Chapitre 2

Étude de quelques protocoles réseau

Nous allons décrire dans ce chapitre quelques protocoles réseau de l'architecture TCP/IP étendue, plus précisément le protocole Ethernet de la couche d'accès, le protocole de résolution d'adresse ARP, les protocoles IPv4 et ICMP de la couche réseau IP, les protocoles UDP et TCP de la couche transport et, enfin, le protocole HTTP de la couche application.

Pour ne pas rester sur une base purement théorique, il est traditionnel depuis [STE–94] d'utiliser un analyseur de trames pour illustrer la description des protocoles. Un **analyseur de trames** (appelé aussi **analyseur réseau** ou **renifleur** pour être proche du terme anglais *sniffer*) permet de récupérer toutes les trames passant par une interface réseau donnée de façon brute à fin d'analyse. Il s'agit des trames partant de l'ordinateur ou arrivant sur cette interface, que celles-ci soient destinées ou non à l'ordinateur. En fait, par défaut, l'interface réseau détruit immédiatement les trames qui ne sont pas destinées à l'ordinateur mais il existe un **mode promiscuité** qui permet de les retenir également. Pour des raisons de sécurité, seul le super-utilisateur peut se servir des analyseurs de trames. Un analyseur de trames peut récupérer (et analyser) les trames en entier ou les premiers octets de celle-ci au choix, ce qui permet de donner des indications sur les en-têtes de protocole sans récupérer l'intégralité des données, ce qui est suffisant pour analyser le réseau et ses problèmes éventuels.

TCP/IP utilise les dénominations suivantes pour les unités d'information :

Application	Message	
Transport	Segment Datagramme	dans le cas connecté dans le cas non connecté
Réseau	Paquet	
Accès	Trame	

trame se disant *frame* en anglais.

2.1 Protocole Ethernet

Le protocole Ethernet a été défini par Xerox. Il a eu tellement de succès que DEC, Intel et Xerox ont défini en 1978 un standard Ethernet à 10 Mbits/s, appelé le standard **DIX** d'après les initiales des protagonistes. Après deux modifications mineures, il est devenu en 1983 la norme IEEE 802.3, puis plus tard une norme ISO.

2.1.1 Adresse physique MAC

Ethernet [IEEE-802.3] et quelques autres protocoles de la couche d'accès utilisent une adresse physique sur 48 bits, appelée **adresse MAC** (*Media Access Control address* en anglais), dont la structure est donnée sur la figure suivante :

1 bit	1 bit	22 bits	24 bits
I/G	U/L	Partie OUI du constructeur	Partie déterminée par le constructeur

— Le bit `I/G`, le moins significatif de l'adresse, est le bit d'adresse d'individu I ou de groupe G. Si ce bit vaut 0, l'adresse est celle d'un individu. S'il vaut 1, le reste du champ de l'adresse identifie une adresse de groupe qui nécessite une résolution supplémentaire.

Lorsqu'une trame est adressée à un groupe, toutes les stations du groupe la reçoivent. Une diffusion de ce type est appelée **diffusion restreinte**, **diffusion multidestinataire** ou **diffusion multidiffusion** (*multicast* en anglais).

— Le deuxième bit, `U/L`, est le bit local L ou universel U. S'il vaut 0, l'adresse est déterminée (en partie) par un organisme international pour assurer l'unicité de l'adresse MAC au niveau mondial. S'il vaut 1, l'adresse a été définie localement et l'interface ne doit pas être reliée à Internet.

L'organisme d'administration universel fut d'abord Xerox (le créateur d'Ethernet) et est actuellement l'IEEE (*Institute of Electrical and Electronics Engineers*).

— Dans le cas où le deuxième bit vaut 0, les 24 premiers bits constituent l'**OUI** (pour *Organization Unique Identifier*), attribuée à chaque constructeur par IEEE. La liste des OUI se trouve dans le fichier :

`http://standards.ieee.org/regauth/oui/oui.txt`

Les 24 derniers bits sont déterminés par le constructeur de l'interface réseau, que l'on peut quelquefois changer localement.

Si la totalité de l'adresse est composée de bits à 1, l'adresse a une signification spéciale : elle indique que toutes les stations du réseau local sont considérées comme destinataires. On parle de **diffusion générale** (*broadcast* en anglais).

Un constructeur de cartes réseau est donc identifé par 22 bits. Ce système autorise donc 2^{22} combinaisons – soit 4 millions environ. Si une organisation est à court d'adresses physiques, l'IEEE a la possibilité de lui affecter un deuxième OUI.

Il se peut, qu'au rythme de croissance actuel d'Internet, on se retrouve à court d'OUI. L'IEEE a défini un format similaire à 64 bits appelé **EUI-64**.

2.1.2 Les trames Ethernet DIX

Commençons par décrire la structure de trame de l'Ethernet d'origine, appelée **trame DIX** (pour *DEC-Intel-Xerox*). Une trame Ethernet ne comprend pas seulement un en-tête et des données. Elle est suivie d'un suffixe comme le montre le schéma suivant :

Préambule	Adresse destinataire	Adresse émetteur	Type	Données	CRC
64 bits	48 bits	48 bits	16 bits	Longueur variable	32 bits

— Le **préambule** est un ensemble de bits dont la fonction principale est de synchroniser le processus de communication et de mettre en évidence tout bruit parasite affectant les premiers bits envoyés.

Dans le cas d'une trame de l'Ethernet d'origine, chacun des 8 octets du préambule contient la séquence binaire `10101010`. D'un point de vue physique, le codage utilisé, dit *codage Manchester*, de cette séquence produit un signal de forme rectangulaire de 10 MHz pendant 6,4 μs, ce qui permet à l'horloge du récepteur de se caler avec celle de l'émetteur. Les parties doivent ensuite rester synchronisées pour le reste de la trame et utiliser le code Manchester pour identifier les délimitations des bits.

Le code Manchester ne nous intéresse pas ici car il est pris en compte de façon matérielle par l'interface réseau et ne concerne donc pas la partie logicielle.

— Les adresses MAC du destinataire et de l'expéditeur suivent.

En fait la norme autorise des adresses sur 2 ou 6 octets, mais les paramètres définis pour la norme à 10 Mbit/s en bande de base n'emploient que des adresses sur 6 octets. Nous ne nous occupons ici que des adresses sur six octets.

— Dans le cas d'une trame DIX, le champ suivant de 16 bits, le **type**, permet au récepteur le démultiplexage au niveau de la couche réseau, autrement dit ce que celui-ci doit faire de la trame. En effet plusieurs protocoles de couche réseau peuvent être employés en même temps sur un même ordinateur donc, lorsqu'une trame Ethernet arrive, le sous-système d'exploitation du réseau doit savoir à quel protocole la remettre. C'est ce champ qui lui permet de le savoir.

Voici quelques valeurs de ce champ :

Valeur décimale	Description
512	Xerox PUP
2048	Internet Protocol (IP)
2051	ECMA
2053	X.25 Level 3
2054	Address Resolution Protocol (ARP)
32821	Reverse ARP
32823	Appletalk

Remarquons que 2048 (IP) et 2054 (ARP) sont les seules valeurs qui intéressent l'architecture TCP/IP.

— Les données suivent l'indicateur de type, dont la longueur varie entre 46 et 1 500 octets, la valeur dépendant du type.

La valeur maximum de 1 500 a été choisie presque de façon arbitraire, principalement parce qu'un transcepteur devait disposer de suffisamment de mémoire pour contenir une trame entière et qu'en 1978 la mémoire RAM était coûteuse.

Outre une longueur maximale, une trame doit respecter une longueur minimale. Bien qu'un champ de données de zéro octet soit parfois utile, cela pose problème. Lorsqu'un

transcepteur détecte une collision, il tronque la trame en cours, si bien que des bits ou des fragments de trame égarés apparaissent continuellement sur le câble. Pour faciliter la distinction entre les trames valides et les rebuts parasites, Ethernet exige que les trames valides possèdent au moins 64 octets, du champ d'adresse de destination au champ de somme de contrôle, ces deux champs inclus. Si le champ des données a une taille inférieure à 46 octets, elles sont complétées par des 0 jusqu'à obtenir 46 octets, ce qui constitue le **champ de remplissage**.

— À la fin de la trame se trouvent quatre octets de vérification, (**CRC** pour *Cyclic Redundancy Check*), qui permet de s'assurer plus ou moins que le contenu de la trame n'a pas été altérée lors de la transmission. Toute passerelle sur la route de transmission doit calculer la valeur de CRC pour la trame et la comparer à la valeur qui figure à la fin de celle-ci. Si les deux sont égales, la trame est envoyée un peu plus loin dans le réseau. Sinon, c'est que la trame a été altérée en cours de route et elle doit donc être détruite (elle sera éventuellement retransmise – après expiration d'un temporisateur (*timer*) – par la machine qui l'a envoyée).

2.1.3 Les trames Ethernet 802.3

Nous avons vu qu'Ethernet, conçu à l'origine par Xerox, est devenu en 1983 un standard IEEE, la plus grande institution professionnelle au monde. Elle publie de nombreuses revues, organise des centaines de conférences par an et dirige une section chargée de développer des normes dans les domaines de l'électronique et de l'informatique. La commission IEEE 802 a notamment standardisé plusieurs types de réseaux locaux.

2.1.3.1 La commission 802 de l'IEEE

Ethernet est le plus célèbre des types de réseaux locaux normalisés par IEEE mais ce n'est pas le seul : il existe aussi, entre autres, l'anneau à jeton (*token ring*), tous les deux maintenant pratiquement abandonnés et surtout Wi-Fi et Bluetooth.

Le tableau ci-dessous présente les groupes de travail dans le cadre du comité IEEE 802 et l'avancement des travaux :

Numéro	**Sujet**	**Avancement**
802.1	Vue d'ensemble sur l'architecture des LAN	
802.2	Contrôle logique de liaison (LLC)	↓
802.3	Ethernet	*
802.4	Bus à jeton	↓
802.5	Anneau à jeton	
802.6	DQDB (premier MAN)	↓
802.7	Technologies à large bande	↓
802.8	Fibre optique	†
802.9	LAN en temps réel	↓
802.10	LAN virtuel (VLAN) et sécurité	↓
802.11	LAN sans fil (Wi-Fi)	*
802.12	AnyLAN de Hewlett-Packard	↓
802.13	Numéro non affecté par superstition	
802.14	Modem câble	↓
802.15	Réseaux personnels (PLAN, Bluetoooth)	*
802.16	Sans fil à large bande	*
802.17	RPR (Resilient Packet Ring)	

* : groupe parmi les plus importants, ↓ : groupe en suspens, † : groupe abandonné.

2.1.3.2 Deux sous-couches de la couche liaison

Le standard IEEE 802 (figure 4.1, [WPRMB–02], p. 136) distingue deux sous-couches dans la couche de liaison des données [IEEE-802] :

— la **sous-couche de contrôle d'accès au canal**, ou **sous-couche MAC** (pour l'anglais *Medium Access Control*), ou encore sous-couche 2a, la plus proche de la couche physique comme son nom le laisse entendre, est chargée de la gestion des accès à un canal partagé.

— La **sous-couche de contrôle de liaison logique** (**LLC** pour *Logical Link Control*), ou sous-couche 2b, dissimule les différences spécifiques aux supports et offre une interface uniforme aux protocoles des couches supérieures. C'est cette sous-couche qui s'occupe du contrôle de flux et des erreurs.

Ce découpage en deux sous-couches présente des avantages :

— Pour transporter des paquets IP, par exemple, aucune garantie de livraison n'est requise ni attendue. On n'a donc pas besoin d'utiliser la sous-couche LLC.

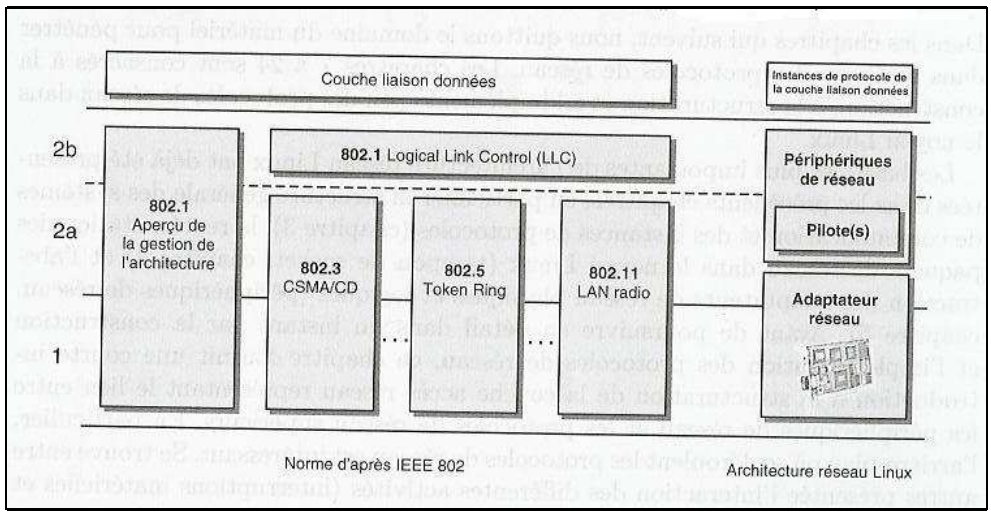

FIGURE 2.1 – Sous-couches MAC et LLC

— La couche LLC est indépendante du type de réseau local : Ethernet ou WiFi.

Lors de l'implémentation :

— L'interface avec la couche 1 (couche physique) et la sous-couche 2a (sous-couche MAC) sont souvent implémentées dans les cartes réseau.

— La sous-couche 2b (LLC) est implémentée dans le sous-système réseau du noyau du système d'exploitation.

2.1.3.3 Protocole 802.3 de la sous-couche LLC

Dans le cas de 802.3, le champ « données » de la trame ne comporte pas d'en-tête de sous-couche 2b ; par contre la signification des champs de l'en-tête Ethernet (*a priori* de la sous-couche MAC) a légèrement changé. On parle alors de **trame Ethernet 802.3**. Il y a deux changements :

— Le premier changement concerne le préambule. Les sept premiers octets restent inchangés ; le huitième octet sert de délimiteur de début de trame (**SFD** pour *Start Frame Delimiter*) afin de rendre le format compatible avec les normes 802.4 (bus à jeton) et 802.5 (anneau à jeton).

— Le second changement a d'abord transformé le champ « type » en champ de **longueur** de trame. Bien sûr, il n'est alors plus possible pour le destinataire de savoir ce qu'il faut faire avec une trame entrante, mais ce problème est résolu par l'ajout d'un petit en-tête dans la portion des données pour apporter cette information.

Malheureusement, à l'époque où la norme 802.3 est publiée, il y a déjà tellement d'équipements et de logiciels en circulation pour Ethernet DIX que peu de fabricants et d'utilisateurs accueillent avec enthousiasme le changement du champ « type » en champ « longueur ». En 1997, l'IEEE jette l'éponge et annonce que les deux formats sont valides. Heureusement, la plupart des champs de type utilisés avant 1997 étaient d'une valeur supérieure à 1 500 (la longueur maximum des données). Par conséquent, une valeur inférieure ou égale à 1 500 peut être considérée comme étant une longueur et une valeur supérieure à 1 500 comme étant un type.

2.2 L'analyseur de trames tcpdump

Le premier analyseur de trames à connaître un grand succès est `tcpdump`, écrit par VAN JACOBSON, Craig LERES et Steve MCCANNE, tous du Laboratoire de Lawrence Berkeley, de l'université de Californie à Berkeley, reposant sur une idée de Jeffrey MOGUL [MOG–90] à Stanford.

2.2.1 Mise en place de tcpdump

2.2.1.1 Mise en place sous Linux ou MacOS

Avec un système d'exploitation totalement compatible UNIX, comme *Linux* ou le MacOS d'*Apple*, l'utilitaire `tcpdump` est en général installé par défaut.

Ouvrir le *Terminal* et taper :

```
>tcpdump
```

on obtient :

```
tcpdump failed
```

ce qui montre, déjà, que l'utilitaire est reconnu.

Cet utilitaire ne pouvant être utilisé que par le superutilisateur, comme nous l'avons déjà dit, il faut taper :

```
>sudo tcpdump
```

pour *SUperuser DO*, dont la première réaction est de demander un mot de passe. Il s'agit du mot de passe du superutilisateur. Rassurez-vous, sur votre ordinateur personnel, il s'agit certainement du même mot de passe que celui de votre compte personnel.

2.2.1.2 Mise en place de WinDump sous *Windows*

L'utilitaire `tcpdump` n'est pas porté pour *Windows* mais il existe un portage compatible constitué de deux parties : une bibliothèque et des pilotes, `WinPcap`, pour récupérer les trames et un analyseur proprement dit, appelé `WinDump`. Ces deux outils sont téléchargeables sur le site :

`http://www.winpcap.org/`

Mise en place de `WinPcap`

Sur la page d'accueil de `WinPcap`, cliquer sur l'icône étiquetée **Download** `Get WinPCap`, qui conduit à la page :

`http://www.winpcap.org/install/default.htm/`

Cliquer sur l'icône de même nom, qui permet de télécharger (et conserver ou non) un fichier exécutable de nom du type `WinPcap_4_1_3.exe`.

Faites exécuter ce fichier ! Suivre les instructions données à lécran ! L'utilitaire d'installation détecte automatiquement le système d'exploitation et installe en conséquence les logiciels corrects.

Lorsque tout s'est bien passé, les applications reposant sur *WinPcap*, ce qui est le cas de *WinDump*, sont prêtes à fonctionner. Ceci ne se voit évidemment pas tant qu'on n'a pas installé par ailleurs une telle application.

Mise en place de `WinDump`

Téléchargement.- Placez-vous sur la page d'accueil de `WinDump` :

`http://www.winpcap.org/windump/`

Ne tenez pas compte du descriptif, duquel on déduit que l'application n'a pas été mise à jour depuis la vénérable époque de *Windows Vista*. Les fonctions en langage C pour la programmation *Windows* n'ont pas fondamentalement changé. Les pilotes de *WinPcap* sont bien ceux de votre système d'exploitation et donc l'application fonctionnera sur celui-ci.

Cliquez sur l'icône étiquetée **Download** `Get WinDump`, qui conduit à la page :

`http://www.winpcap.org/windump/install/default.htm/`

Cliquez sur l'icône de même nom, qui permet de télécharger et conservez, cette fois-ci, le fichier exécutable de nom `WinDump.exe` dans un répertoire adéquat (par exemple `h:\Programmes\Windump`).

Utilisateur en tant qu'administrateur.- Ouvrez une occurrence de l'*Invite de commandes*, placez-vous dans le répertoire dans lequel se trouve *WinDump.exe* et appelez cet exécutable.

Il y a juste un retour au prompteur. C'est déjà un bon signe : cela veut dire que le système a reconnu l'exécutable. Mais il ne sert pas à grand chose si rien n'en ressort.

En appuyant sur l'icône de l'*Invite de commandes* avec le bouton de droit de la souris, choisissez, non pas « Ouvrir », mais « Exécuter en tant qu'administrateur ». Appelez, dans cette occurrence, l'exécutable `WinDump.exe` : on va, cette fois-ci, un peu plus loin puisqu'on voit s'afficher quelque chose comme « *windump : listening on...* ». Il y a donc bien eu une réaction ; nous verrons comment continuer ci-dessous.

Changement de la variable *Path*.- Rien n'empêche de se placer systématiquement dans le répertoire dans lequel se trouve l'exécutable pour l'exécuter, puisque nous n'enregistrerons rien en général. On préfère cependant pouvoir l'exécuter depuis n'importe quelle position de répertoire de l'*Invite de commandes*. On sait qu'il faut, pour cela, modifier la valeur de la variable d'environnement appelé *Path*.

La méthode habituelle, utilisant l'interface graphique, risque de ne pas fonctionner. En effet, on change ainsi la variable *Path* pour l'utilisateur et non pas pour l'administrateur.

Une autre façon de faire est d'utiliser la ligne de commande et non pas l'interface graphique. Pour cela, dans une *Invite de commandes* en tant qu'administrateur (et non en tant qu'utilisateur), exécutez la commande :

`Setx /M PATH "%PATH%;h:\programmes\Windump"`

en remplaçant, si nécessaire le chemin adéquat du répertoire.

Si tout s'est bien passé, on obtient le comportement ci-dessus (avec réaction) depuis n'importe quelle occurrence de répertoire dans une instance d'*Invite de commandes* en tant qu'administrateur.

2.2.2 les options de tcpdump

Nous n'avons pas obtenu grand chose avec l'appel de `WinDump` (ou de `tcpdump`) car cet utilitaire exige des *options* de commande pour afficher quelque chose. Toute option commence par un tiret « - » suivi d'une lettre, éventuellement suivi d'un *paramètre*.

Liste des options.- L'option « `-?` » donne la liste des options possibles (évidemment sous forme très concise ; il s'agit d'un rappel et non d'explications détaillées) :

```
C:\>windump -?
windump version 3.9.5, based on tcpdump version 3.9.5
WinPcap version 4.1.3 (packet.dll version 4.1.0.2980), based on libpcap version
1.0 branch 1_0_rel0b (20091008)
Usage: windump [-aAdDeflLnNOpqRStuUvxX] [ -B size ] [-c count] [ -C file_size ]
                [ -E algo:secret ] [ -F file ] [ -i interface ] [ -M secret ]
                [ -r file ] [ -s snaplen ] [ -T type ] [ -w file ]
                [ -W filecount ] [ -y datalinktype ] [ -Z user ]
                [ expression ]

C:\>
```

Les interfaces réseau disponibles.- Chaque ordinateur dispose en général de plusieurs interfaces réseau : l'interface `loop` qui désigne l'ordinateur lui-même, une interface Ethernet, une interface WiFi, etc.

Pour connaître ces interface, et surtout le numéro qui lui a été attribué, on utilise l'option « `-D` », ce qui donne, par exemple :

```
C:\>windump -D
1.\Device\NPF_{D6E7B08F-97D4-4BA3-A281-A966C377E5F8} (Realtek PCIe GBE Family Co
ntroller)
2.\Device\NPF_{942950F4-6CB1-4BCF-9D17-F650BFDBF6C7} (Microsoft)
3.\Device\NPF_{D9057061-41BF-4B77-AB67-5D2A8A8AEC55} (Microsoft)

C:\>
```

Il s'agit ici d'une interface Ethernet et de deux interfaces WiFi, même si ce n'est pas clair.

Choix de l'interface réseau à écouter.- Sur un ordinateur de bureau, il y a des chances pour que les trames passent le plus fréquemment par l'interface Ethernet. Mettons-nous donc à l'écoute de celle-ci :

```
C:\>windump -i 1
windump: listening on \Device\NPF_{D6E7B08F-97D4-4BA3-A281-A966C377E5F8}
11:23:13.578930 IP6 Patrick-PC.1900 > Patrick-PC.58615: UDP, length 391
11:23:13.789879 arp who-has 192.168.27.1 tell 192.168.27.14
11:23:14.403934 IP
3 packets captured
52 packets received by filter
0 packets dropped by kernel

C:\>
```

Il y a écoute à partir du moment où on appuie sur la touche « Entrée ». Pour l'arrêter, on tue le processus en appuyant sur « CTRL-C ».

Dans ce cas, trois *paquets* ont été capturés et analysés rapidement, un par ligne, en commençant par l'heure à laquelle le paquet a été récupéré.

Limitation du nombre de trames à analyser.- L'option « `-c` » permet de limiter le nombre de trames à récupérer (et à analyser), ce qui évite d'avoir recours à la façon brutale de terminer le processus :

```
C:\>windump -i 1 -c 5
windump: listening on \Device\NPF_{D6E7B08F-97D4-4BA3-A281-A966C377E5F8}
11:47:02.305845 IP6 Patrick-PC.58615 > ff02::c.1900: UDP, length 146
11:47:02.306258 IP6 Patrick-PC.58616 > ff02::c.1900: UDP, length 146
11:47:02.801830 IP6 Patrick-PC.1900 > Patrick-PC.58616: UDP, length 392
11:47:03.832422 IP frontcmc01-ext.u-pec.fr.443 > Patrick-PC.62842: P 2927855207
2927855244(37) ack 2222779102 win 509
11:47:03.832714 IP frontcmc01-ext.u-pec.fr.443 > Patrick-PC.62842: P 37:90(53)
ck 1 win 509
5 packets captured
8 packets received by filter
0 packets dropped by kernel

C:\>
```

Récupération des trames.- En fait nous voulons ici récupérer les trames, que nous analyserons « à la main ». Utilisons pour cela l'option « `-XX` » :

```
C:\>windump -i 1 -c 3 -XX
windump: listening on \Device\NPF_{D6E7B08F-97D4-4BA3-A281-A966C377E5F8}
11:51:41.208338 IP Patrick-PC.58618 > 239.255.255.250.1900: UDP, length 133
        0x0000:  0100 5e7f fffa 10bf 484c b6a3 0800 4500  ..^.....HL....E.
        0x0010:  00a1 51f8 0000 0111 b57f c0a8 0132 efff  ..Q..........2..
        0x0020:  fffa e4fa 076c 008d b273 4d2d 5345 4152  .....l...sM-SEAR
        0x0030:  4348 202a 2048 5454 502f 312e 310d 0a48  CH.*.HTTP/1.1..H
        0x0040:  6f73 743a 3233 392e 3235 352e 3235 352e  ost:239.255.255.
        0x0050:  3235 303a 3139 3030 0d0a 5354 3a75 726e  250:1900..ST:urn
11:51:41.208926 IP FREEBOX.1900 > Patrick-PC.58618: UDP, length 302
        0x0000:  10bf 484c b6a3 0024 d4ac 22a9 0800 4500  ..HL...$.."...E.
        0x0010:  014a 1195 4000 4011 a38d c0a8 01fe c0a8  .J..@.@.........
        0x0020:  0132 076c e4fa 0136 bc2a 4854 5450 2f31  .2.l...6.*HTTP/1
        0x0030:  2e31 2032 3030 204f 4b0d 0a53 4552 5645  .1.200.OK..SERVE
        0x0040:  523a 204c 696e 7578 2f32 2e36 2055 506e  R:.Linux/2.6.UPn
        0x0050:  502f 312e 3020 6662 7869 6764 642f 312e  P/1.0.fbxigdd/1.
11:51:41.209297 IP 192.168.27.14.32801 > 239.255.255.250.1900: UDP, length 133
        0x0000:  0100 5e7f fffa 0024 d4ac 22a9 0800 4500  ..^....$.."...E.
        0x0010:  00a1 2aaa 4000 0111 82f1 c0a8 1b0e efff  ..*.@...........
        0x0020:  fffa 8021 076c 008d 05ac 4d2d 5345 4152  ...!.l....M-SEAR
        0x0030:  4348 202a 2048 5454 502f 312e 310d 0a48  CH.*.HTTP/1.1..H
        0x0040:  6f73 743a 3233 392e 3235 352e 3235 352e  ost:239.255.255.
        0x0050:  3235 303a 3139 3030 0d0a 5354 3a75 726e  250:1900..ST:urn
```

```
3 packets captured
85 packets received by filter

C:\>
```

Remarquons que, pour chaque trame récupérée, nous avons l'analyse rapide de celle-ci, suivie du contenu (du début) de la trame présentée à la façon d'un éditeur hexadécimal.

2.3 Mise en place de Wireshark

Bien que le premier analyseur de trames à avoir connu un grand succès soit `tcpdump`, on utilise la plupart du temps de nos jours `wireshark` (le requin des fils, connu au début sous le nom d'`ethereal`), qui présente une interface graphique très conviviale et qui analyse de nombreux protocoles. Sa page d'accueil indique d'ailleurs qu'il peut être utilisé à titre pédagogique :

`http://www.wireshark.org/`

Le menu de téléchargement donne accès à des binaires pour Windows (32 ou 64 bits) ainsi que le code source pour les autres systèmes d'exploitation.

2.3.1 Installation

FIGURE 2.2 – Fenêtre d'accueil de Wireshark

2.3.1.1 Cas de Windows

Il suffit de télécharger la dernière version (1.2.1 en septembre 2009 de 17,4 Mo ; 77,3 Mo une fois décompressé) et de l'exécuter (en tant que super-utilisateur).

2.3.1.2 Cas de Linux

`Wireshark` est présent sur la plupart des distributions. Il suffit donc de demander d'installer ce logiciel.

2.3.2 Utilisation

Sous Windows, l'appel en cliquant sur l'icône de `wireshark.exe` fait apparaître la fenêtre de la figure 2.2. Sous *Linux*, il suffit de taper `wireshark &` en ligne de commande, ce qui donne accès directement à ce que l'on voit sur la figure 2.3.

En cliquant sur le nom de l'interface réseau désirée de la liste d'interface apparaissant dans la fenêtre `Capture`, on fait apparaître la fenêtre de la figure 2.3.

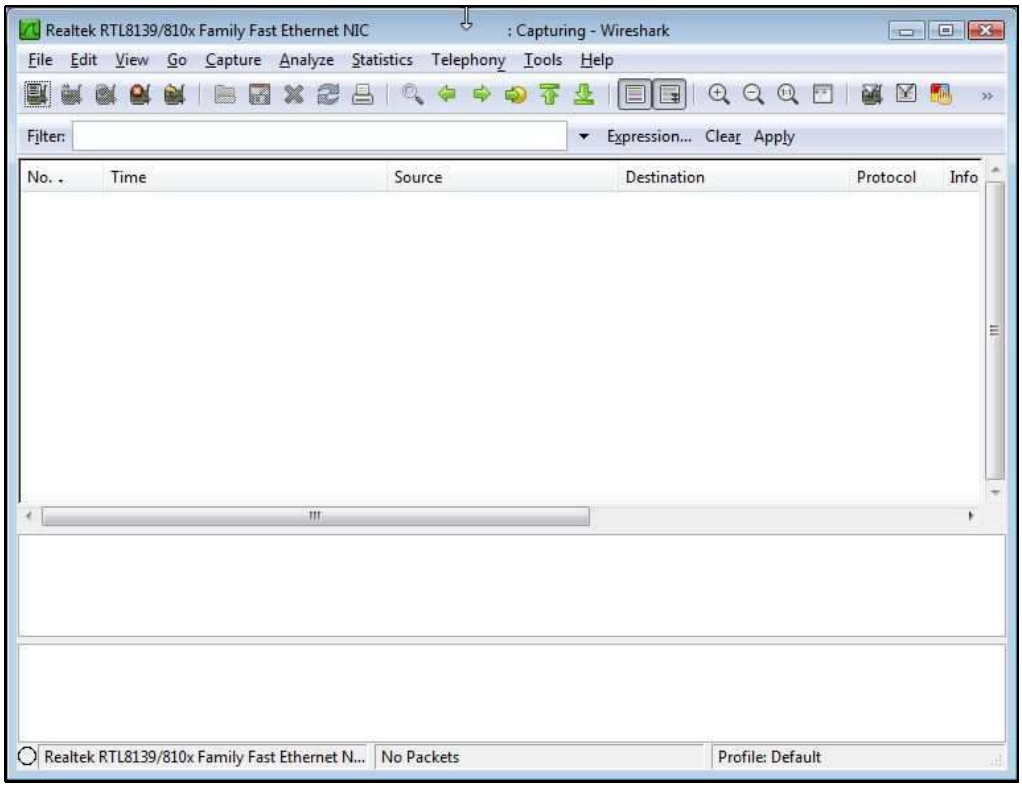

FIGURE 2.3 – Fenêtre d'annonce de capture de Wireshark

Mon ordinateur est relié à Internet grâce à un routeur, relié lui-même à mon fournisseur d'accès Internet (FAI) par la fibre optique. Sur mon réseau local ainsi constitué, l'adresse IP du routeur est classiquement 192.168.1.1 et celle de mon ordinateur 192.168.1.12.

Sous Windows, utilisons `putty` pour effectuer un `telnet`, comme montré sur la figure 2.4.

La fenêtre de capture indique alors que que deux trames (au moins) ont été capturées. Appuyons sur le bouton `arrêter` de celle-ci. La première fenêtre prend alors l'aspect de la figure 2.5.

On y voit trois zones :

— Dans une première zone est affichée la liste des trames capturées avec un numéro de trame, l'heure à laquelle elle a été capturée (par défaut depuis le lancement de la capture mais on peut demander l'heure réelle), l'adrese réseau source de la trame (adresse IP en général, mais pas toujours), l'adresse réseau de la destination, le protocole de plus haut niveau détecté et quelques autres informations.

— Dans une deuxième zone le nombre d'octets de la trame ainsi que le nombre d'octets cap-

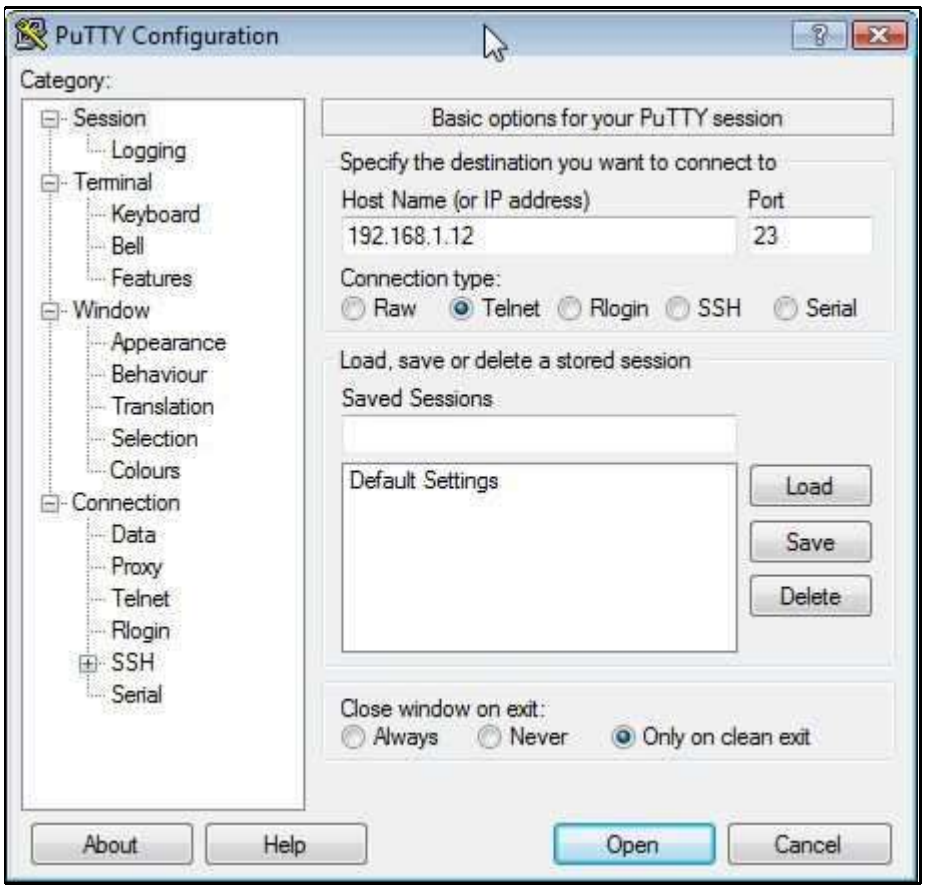

FIGURE 2.4 – Utilisation de Putty

turés sont indiqués, suivi du détail des en-têtes de tous les protocoles jusqu'au protocole final (ici Ethernet et ARP).

— Dans la troisième zone apparaissent les octets capturés de la trame. Chaque ligne contient seize octets. Elle commence par le numéro (en hexadécimal) du premier octet de la ligne, suivi de deux groupes de huit octets en hexadécimal, suivi du caractère ASCII correspondant (remplacé par un point si celui-ci n'est pas affichable).

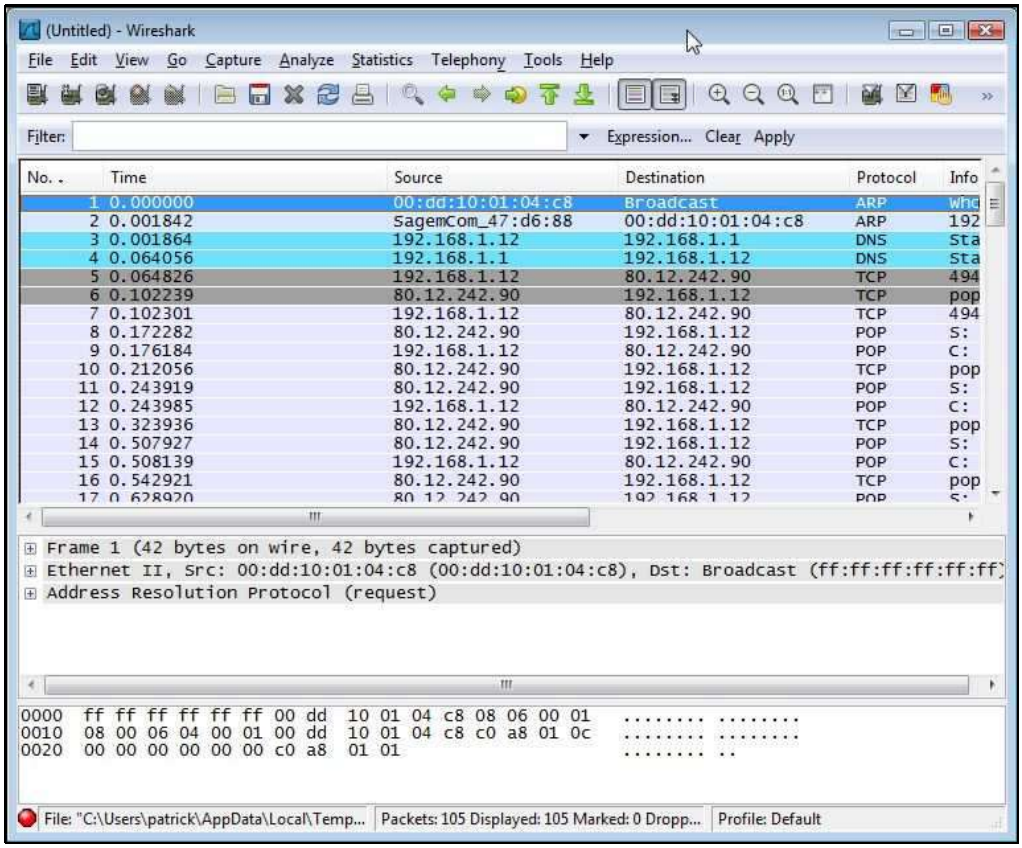

FIGURE 2.5 – Liste des trames capturées

2.4 Analyse des en-têtes Ethernet

Cliquons sur le petit carré contenant un signe plus au début de ligne Ethernet de la deuxième zone. On obtient l'aspect de la figure 2.6 : d'une part on a plus de renseignements sur l'en-tête de la trame ; d'autre part les octets de la trame concernant cet en-tête sont surchargés en bleu foncé (que ce soit en hexadécimal ou en ASCII).

Les trames transitant sur le support physique sont récupérées par l'interface réseau (une carte Ethernet). Le préambule et le champ CRC ne sont pas placés dans le tampon de l'interface (puisqu'ils sont censés correspondre à ce qu'il faut) et donc ne sont pas récupérés par le sous-système réseau du système d'exploitation.

— L'en-tête de la trame commence donc par les 48 bits (ou six octets) de l'adresse MAC du destinataire. Il s'agit ici d'un envoi à toutes les interfaces du réseau local. Il est traditionnel de noter les adresses MAC en hexadécimal, chaque octet étant séparé du suivant par deux points. Il n'est pas difficile de traduire cette adresse sous sa forme traditionnelle : `ff:ff:ff:ff:ff:ff`.

— Cette adresse MAC est suivie des 48 bits (ou six octets) de l'adresse MAC de l'expéditeur. Il s'agit ici de notre ordinateur, le seul qui soit sur notre réseau local (à part le routeur). Ce n'est pas toujours le cas dans un vrai réseau local avec plusieurs hôtes lorsqu'on utilise le mode promiscuité : on peut récupérer et analyser des trames qui ne nous sont pas

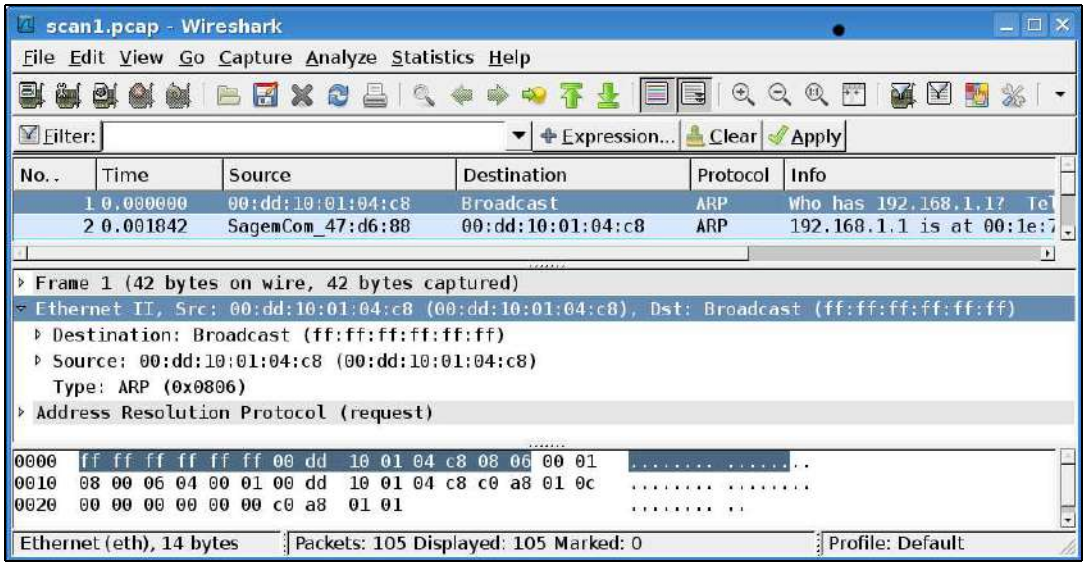

FIGURE 2.6 – En-tête Ethernet

destinées ou que nous n'avons pas envoyées. L'adresse MAC de notre carte Ethernet est donc `00:dd:10:01:04:c8`.

Une recherche dans le document `oui.txt` montre que l'adresse du constructeur n'y est pas référencée. Il s'agit d'une carte chinoise dont le constructeur *realteck* n'a visiblement pas demandé d'OUI.

— Les seize bits (ou deux octets) suivants indiquent le type du protocole de la couche supérieure. On a ici `0806h`, soit 2054, qui spécifie le protocole réseau ARP.

— Les reste est constitué des données Ethernet, commençant par l'en-tête ARP sur lequel nous allons revenir dans la section suivante.

2.5 Le protocole de résolution d'adresse ARP

2.5.1 Le problème

Les en-têtes correspondant à chaque protocole contiennent un certain nombre de paramètres. Le paramètre le plus important pour la couche d'accès est l'**adresse physique** qui permet de déterminer à quel ordinateur du réseau local la trame est destinée : il s'agit de l'adresse MAC dans le cas d'Ethernet. Le paramètre le plus important pour la couche réseau est l'**adresse réseau** qui permet de déterminer à quel ordinateur de l'interréseau le paquet est destiné : il s'agit de l'adresse IP dans le cas de IP, plus précisément de l'adresse IPv4 dans le cas de IPv4.

Lorsque mon routeur reçoit un paquet destiné à mon réseau local, celui porte l'adresse réseau de destination. Par contre le protocole de la couche d'accès (celui utilisé par le réseau local) ne connaît que les adresses physiques. Il faut donc effectuer une conversion entre adresses réseau des équipements reliés à mon réseau local et adresses physiques de ces mêmes équipements.

Ceci s'effectuait à l'aide de tables statiques placées dans chaque ordinateur, ou tout au moins au niveau du roteur du réseau local, au début de l'ARPAnet. Toutefois le développement de l'ARPAnet a rendu cette méthode trop rigide et trop onéreuse en mémoire. C'est pourquoi la [RFC 826] a introduit l'ARP (*Address Resolution Protocol*) en vue de la conversion des adresses.

Bien que la famille de protocoles TCP/IP soit devenue la norme de premier plan pour la presque totalité des réseaux informatiques, l'ARP n'a pas seulement été mis au point pour la conversion spécifique entre adresses IPv4 et adresses MAC.

2.5.2 Adresse IP

2.5.2.1 Définition

IPv4 utilise une adresse sur 32 bits pour identifier une machine sur un interréseau, c'est-à-dire des réseaux locaux interconnectés, appelée **adresse IP** ou **adresse Internet**.

Les adresses IP sont attribuées e puis 1998 par l'**ICANN** (*Internet Corporation for Assigned Names and Numbers*, soit Société pour l'attribution des noms de domaine et des numéros sur Internet). Un réseau qui n'est pas connecté à Internet peut déterminer son propre adressage mais, pour tous les accès Internet, l'adresse doit être enregistrée auprès de l'ICANN pour éviter les doublons.

Plus exactement, l'**IANA** (*Internet Assigned Numbers Authority*), créée avant l'ICANN mais qui en constitue désormais un département, a découpé l'espace d'adressage IPv4 en 256 blocs /8. Chacun de ces blocs est libre, réservé, assigné dans le passé ou alloué à l'un des cinq **RIR** (*Regional Internet Registry* ou *Registre Internet Régional*), par ordre de création :

- **RIPE-NCC** (*Réseaux IP Européens*, créé en 1992) pour l'Europe et le Moyen-Orient ;
- **APNIC** (*Asia Pacific Network Information Center*, créé en 1993) pour l'Asie et le Pacifique ;
- **ARIN** (*American Registry for Internet Numbers*, créé en 1997) pour l'Amérique du Nord (entre 1993 et 1997, ce rôle était attribué à InterNIC [*The Internet's Network Information Center*]) ;
- **LACNIC** (*Latin America and Caribbean Network Information Center*, créé en 1999) pour l'Amérique latine et les îles des Caraïbes ;
- **AfriNIC** (*African Network Information Center*, créé en 2005) pour l'Afrique.

À leur tour, les RIR distribuent des blocs d'adresses à des « registres Internet locaux » (en anglais *Local Internet Registries* ou **LIR**) qui les distribuent aux utilisateurs finaux dans leur zone d'opration. Ce représentant est chargé d'attribuer les adresses dans cette zone, par exemple **Afnic** (*Association française pour le nommage Internet en coopération*) :

`http://www.afnic.fr`
pour la France.

2.5.2.2 Représentation utilisateur

Puisqu'il est difficile de lire un nombre de 32 bits, on représente souvent ces nombres dans les applications sous la forme de quatre nombres décimaux de 8 bits, séparés par des points, par exemple :

127.40.8.72

127 représentant dans ce cas l'octet de poids fort.

2.5.3 Requête, réponse et cache ARP

Lorsqu'un ordinateur (ou équipement) A veut envoyer un paquet à l'ordinateur (ou routeur) B, situé dans le même réseau local et dont il connaît l'adresse réseau mais pas l'adresse physique, il envoie une demande (**requête ARP**, *request* en anglais) à tous les ordinateurs du réseau local (en diffusion générale, c'est-à-dire avec l'adresse MAC `FF:FF:FF:FF:FF:FF` dans le cas d'Ethernet). L'ordinateur recherché reconnaît à l'adresse réseau placée dans le paquet ARP que la requête lui est destinée et envoie une **réponse ARP** (*reply* en anglais) à l'ordinateur demandeur A, lui communiquant alors son adresse physique.

Pour ne pas être contraint de demander sans arrêt l'adresse physique lors des paquets suivants, disons dans les quelques secondes qui suivent, A enregistre l'adresse physique de B dans une table locale, appelée **cache ARP**. L'ordinateur B peut également extraire l'adresse physique de l'ordinateur A à partir de la requête et l'enregistrer à titre conservatoire dans son cache ARP.

2.5.4 Structure des messages ARP

Les requêtes et les réponses ARP ont une structure identique. Elles se différencient par le champ `opération`. Cette structure est la suivante :

<table>
<tr><td>0 7</td><td>8 15</td><td>16 31</td></tr>
<tr><td colspan="2">Type matériel</td><td>Type de protocole</td></tr>
<tr><td>Longueur d'adresse couche 2 (n)</td><td>Longueur d'adresse couche 3 (m)</td><td>Opération</td></tr>
<tr><td colspan="3">Adresse de l'émetteur (couche 2) : n octets</td></tr>
<tr><td colspan="3">Adresse de l'émetteur (couche 3) : m octets</td></tr>
<tr><td colspan="3">Adresse cible (couche 2) : n octets</td></tr>
<tr><td colspan="3">Adresse cible (couche 3) : m octets</td></tr>
</table>

— Le **type matériel** spécifie le protocole de la couche 2 utilisé. Il s'agit par exemple de la valeur 1 pour le protocole Ethernet.

— Le **type de protocole** spécifie le protocole de la couche 3 utilisé, par exemple `0x08 00` pour IPv4.

— La **longueur de l'adresse de la couche 2** spécifie la longueur n, en octets, de l'adresse de la couche 2 pour le protocole utilisé. Dans le cas d'une adresse MAC de 48 bits, on a donc $n = 6$.

— La **longueur de l'adresse de la couche 3** spécifie la longueur m, en octets, de l'adresse de la couche 3 pour le protocole utilisé. Dans le cas d'une adresse IPv4 de 32 bits, on a donc $m = 4$.

— Le champ **opération** indique le type de l'unité de données de protocole ARP : 1 pour une requête ARP, 2 pour une réponse ARP.

— Les champs **adresse de la couche 2 de l'expédieur** et **adresse de la couche 2 du destinataire** sont constitués chacun de n octets.

— Les champs **adresse de la couche 3 de l'expédieur** et **adresse de la couche 3 du destinataire** sont constitués chacun de m octets.

Ces commandes sont encapsulées entre un en-tête et un suffixe de couche 2 et constituent ainsi une trame qui est envoyée en diffusion générale.

Exemple. L'ordinateur A, d'adresse MAC `49:72:16:08:64:14` et d'adresse IP `129.25.10.72`, qui veut rechercher l'adresse MAC de l'ordinateur d'adresse IP `129.25.10.11`, envoie l'unité suivante en diffusion générale, c'est-à-dire à `FF:FF:FF:FF:FF:FF` :

0 7	8 15	16 31
0x00 01		0x08 00
6	4	0x00 01
49 72 16 08 64 14		
129 25 10 72		
00 00 00 00 00 00		
129 25 10 11		

L'ordinateur B, d'adresse MAC `49:78:21:21:23:90` et d'adresse IP `129.25.10.11`, répond alors à `49:72:16:08:64:14`, c'est-à-dire à l'ordinateur A :

0 7	8 15	16 31
0x00 01		0x08 00
6	4	0x00 02
49 72 16 08 64 14		
129 25 10 72		
49 78 21 21 23 90		
129 25 10 11		

2.5.5 Analyse des messages ARP

2.5.5.1 Cas d'une requête

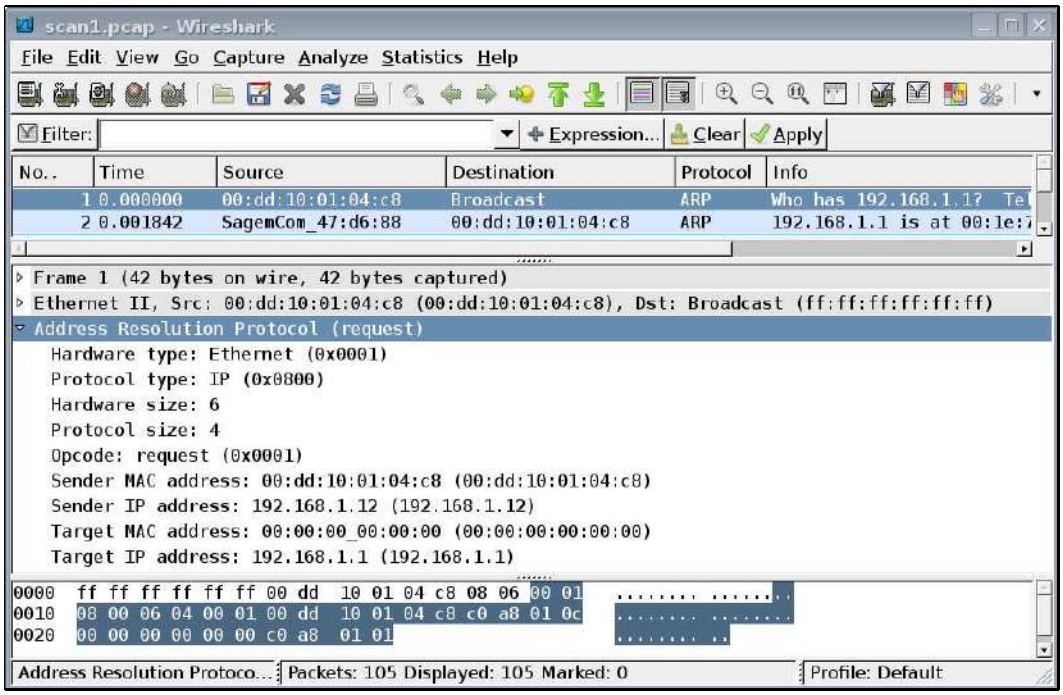

FIGURE 2.7 – Requête ARP

Nous avons vu que la première trame capturée ci-dessus est un message ARP (figure 2.6). Analysons-le. Nous avons ce qui apparaît à la figure 2.7.

— L'en-tête de la trame Ethernet a déjà été étudié ci-dessus. Passons donc aux données de cette trame, qui contient le message ARP.

— Les deux premiers octets indiquent le type de la couche d'accès : il s'agit ici de 1 pour Ethernet. Les deux octets suivants indiquent le type de la couche réseau : il s'agit ici de `800h`, ou 2 048, pour le protocole IP (certainement IPv4 mais il peut y avoir un doute).

— L'octet suivant indique la longueur d'une adresse de la couche d'accès : on a ici 6 octets, soit 48 bits, ce qui est bien la longueur d'une adresse MAC. L'octet suivant indique la longueur d'une adresse de la couche réseau : on a ici 4 octets, soit 32 bits, ce qui est la longueur d'une adresse IPv4.

— Les deux octets suivants indiquent le type d'opération : puisque c'est 1, il s'agit d'une requête ARP.

— Les six octets (d'après ce que nous avons déterminé ci-dessus) suivants spécifient l'adresse physique de l'expéditeur : on retrouve ici l'adresse MAC de notre ordinateur. Les quatre octets (d'après ce que nous avons déterminé ci-dessus) suivants spécifient l'adresse IP de l'expéditeur : on retrouve ici l'adresse IP de notre ordinateur.

— Les six octets suivants spécifient l'adresse MAC que nous recherchons : elle est donc initialisée à zéro. Les quatre octets suivants spécifient l'adresse IP de l'hôte dont on recherche

l'adresse MAC : on trouve ici l'adresse IP de ce qui est certainement le routeur pour ce réseau (192.168.1.1).

2.5.5.2 Cas de la réponse

La deuxième trame de notre capture, qui apparaît à la figure 2.8, constitue la réponse.

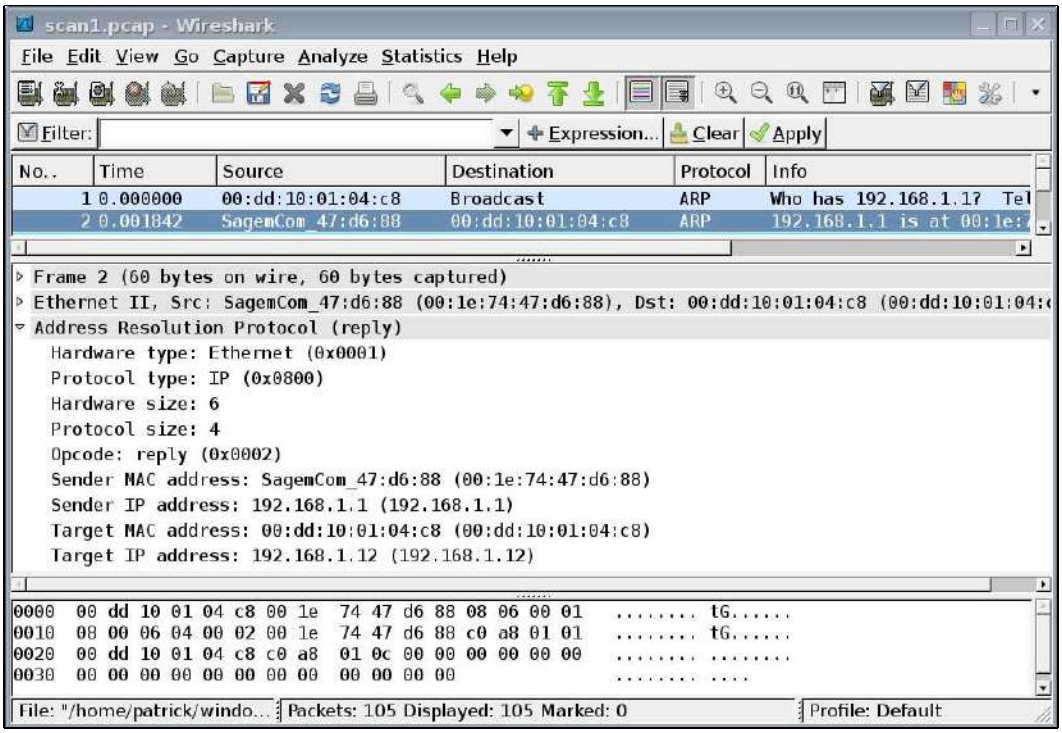

FIGURE 2.8 – Réponse ARP

— Le type de l'en-tête Ethernet est, comme pour la trame précédente, `0806h`, soit 2 054, pour le protocole ARP. Les données de la trame constituent certainement le message ARP ; en fait nous verrons qu'il est suivi d'autre chose.

— Les deux premiers octets du message ARP indiquent le type de la couche d'accès : il s'agit ici de 1 pour Ethernet. Les deux octets suivants indiquent le type de la couche réseau : il s'agit ici de `800h`, ou 2 048, pour le protocole IP.

— L'octet suivant indique la longueur d'une adresse de la couche d'accès : on a ici 6 octets, soit 48 bits, ce qui est bien la longueur d'une adresse MAC. L'octet suivant indique la longueur d'une adresse de la couche réseau : on a ici 4 octets, soit 32 bits, ce qui est la longueur d'une adresse IPv4.

— Les deux octets suivants indiquent le type d'opération : il s'agit ici de 2 pour une réponse ARP.

— Les six octets (d'après ce que nous avons déterminé ci-dessus) suivants spécifient l'adresse MAC de l'expéditeur : il s'agit de l'adresse MAC recherchée. Les quatre octets (d'après ce que nous avons déterminé ci-dessus) suivants spécifient l'adresse IP de l'expéditeur.

— Les six octets suivants spécifient l'adresse MAC du destinataire : on retrouve l'adresse MAC de notre ordinateur. Les quatre octets suivants spécifient l'adresse IP du destinataire : on retrouve ici l'adresse IP de notre ordinateur.

— Nous ne sommes pas arrivés au bout de la trame, contrairement à ce qui se passait dans le cas précédent. Remarquons d'ailleurs que la taille de la trame capturée est de soixante octets alors que la trame précédente était de quarante-deux octets.

Ceci est dû au fait que la longueur minimale d'une trame Ethernet (sans préambule et sans CRC) est de 60 octets pour éviter de prendre en compte les débuts de trame qui subissent une collision. Les 18 derniers octets, tous à zéro, constituent le **remplissage** (*trailer* en anglais).

Le champ de remplissage de la trame précédente était vide car la trame a été récupérée avant d'être envoyée à l'interface réseau, or c'est celle-ci qui ajoute le remplissage éventuel.

2.6 Le protocole de couche réseau IPv4

2.6.1 Étude générale de la couche réseau

2.6.1.1 Le rôle fondamental de la couche réseau : le routage

Notion.- La couche réseau gère le déplacement des paquets dans un interréseau, c'est-à-dire sur des machines qui ne se trouvent pas nécessairement dans le même réseau local. Elle est donc responsable du **routage** (*routing* en anglais) des paquets, c'est-à-dire de la détermination du chemin à suivre par le paquet : il est nécessaire de déterminer le prochain **saut** (*hop* en anglais), c'est-à-dire la première machine de la route vers la destination à travers laquelle va passer le paquet. Il peut s'agir de la machine elle-même, d'une autre machine du réseau local, de la machine de destination ou d'un **routeur** (*router* en anglais), c'est-à-dire d'une machine *ad hoc* spécialisée dans la détermination de la route.

Routeur.- Un routeur comprend au moins deux interfaces réseau et en moyenne quatre à cinq. Le routage comprend deux modules :

- La **redirection** (*forwarding* en anglais) recherche, pour chaque paquet, à partir de la destination de celui-ci dans une table l'interface sur laquelle transmettre ce paquet. Cette table est appelée **table de redirection** (*forwarding table* en anglais).
- La tâche de routage proprement dit consiste à construire et à maintenir la table de redirection.

Routage et adresse IP.- L'algorithme de routage, dont nous ne parlerons pas, s'appuie sur la structure régionale des adresses IP : le routage serait beaucoup plus difficile à effectuer si on utilisait les adresses physiques.

Les adresses IP sont structurées de manière hiérarchique et sont formées de deux parties distinctes : une **adresse de sous-réseau** (*network part* en anglais) et un **identifiant de système terminal** (*host part* en anglais). L'adresse de sous-réseau est identique pour tous les éléments présents dans un sous-réseau donné. Quant à l'identifiant de système terminal, il est destiné à distinguer les différents éléments au sein d'un sous-réseau donné.

En ce qui concerne le routage, l'identifiant du système terminal peut être ignoré jusqu'à ce que le paquet soit parvenu dans le bon sous-réseau. De ce fait, les routeurs n'ont pas besoin de connaître les identifiants des systèmes terminaux, ce qui permet ainsi de réduire de manière notable la quantité d'informations à enregistrer, grâce au découpage de l'adresse IP en deux parties. La partie sous-réseau d'une adresse IP se trouve au début, d'où le synonyme **préfixe réseau** que l'on rencontre quelquefois.

On utilise actuellement sur Internet le système **CIDR** (*Classless Inter-Domain Routing*), défini dans [RFC 1518] et [RFC 1519].

Les préfixes réseau ont une longueur quelconque. L'information concernant la longueur de l'identifiant du sous-réseau doit être transmise en plus de l'adresse réseau. On a le choix pour cela entre deux notations utilisateur :

- Dans la première notation, le nombre de bits correspondant au préfixe du sous-réseau est indiqué sous forme d'un nombre décimal, séparé de l'adresse par une barre oblique. Ainsi, l'adresse **192.168.152.0/21** désigne un sous-réseau dont le préfixe est représenté par les 21 premiers bits de l'adresse IP.
- La seconde notation consiste à ajouter à l'adresse IP un masque de bits de même longueur, le **masque réseau**, dans lequel tous les bits appartenant au préfixe du sous-réseau dans l'adresse IP ont la valeur 1. Par exemple le réseau évoqué ci-dessus est noté :

192.168.152.0/255.255.248.0.

2.6.1.2 Éviter la congestion du réseau : la durée de vie d'un paquet

Lorsqu'on envoie un paquet d'une machine A à une machine B, rien ne dit, d'une part, que la machine B existe et, d'autre part, que l'algorithme de routage soit suffisamment performant pour que cette machaine B, si elle existe, soit trouvée en peu de temps. Le paquet risque donc de parcourir le réseau très longtemps, ou même indéfiniment, au risque de le congestionner. Il faut donc un mécanisme pour éviter de phénomène de survenir.

Pour IPv4, un champ **durée de vie** (`TTL` pour l'anglais *Time To Live*), de taille un octet, est placé dans l'en-tête. Il s'agit d'un compteur qui indique le nombre maximal de systèmes intermédiaires (routeurs ou plutôt sauts) que le paquet peut traverser avant d'être détruit par le nœud en cours avec, dans ce cas, un message ICMP envoyé à la machine émettrice (*Time exceeded*).

La durée de vie est fixée initialement par le nœud émetteur lors de l'assemblage du paquet. Elle est décrémentée au passage de chaque saut.

2.6.1.3 Un rôle auxiliaire de la couche réseau : la fragmentation

Lorsqu'un équipement a déterminé le saut suivant, grâce à l'algorithme de routage, il doit ensuite se préoccuper de la capacité du support physique pour aller jusqu'à ce saut.

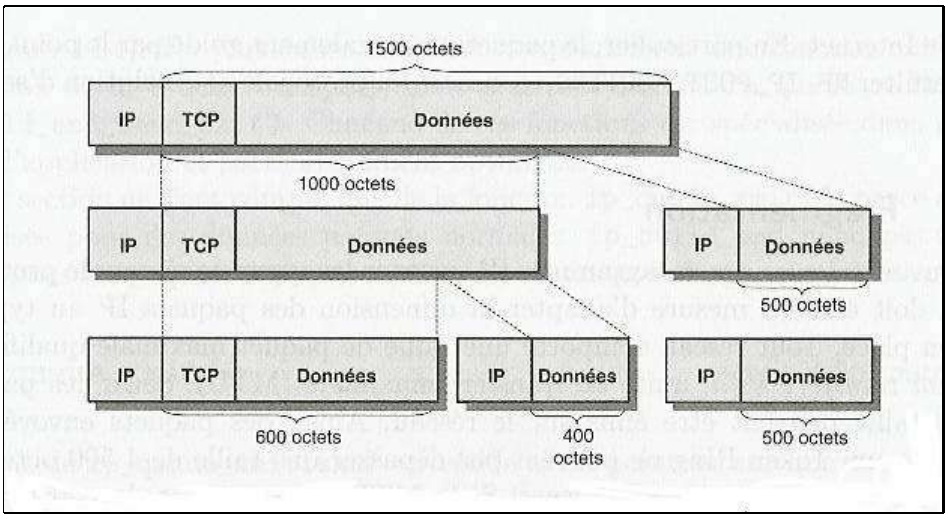

FIGURE 2.9 – Fragmentation d'un paquet IP

Pour pouvoir envoyer des paquets IP sur tous les types de réseau physique, le protocole IP doit être en mesure d'adapter la dimension de ceux-ci au type de réseau en place. Tout support physique a une taille de trame maximale, appelée **MTU** (pour l'anglais *Maximum Transfert Unit*, unité de transfert maximale). Seules des trames inférieures à cette taille peuvent être envoyées sur le support physique.

Si la MTU d'un support de transfert est inférieure à la taille d'un paquet à expédier, ce dernier doit être divisé en paquets IP plus petits, appelés **fragments**, grâce à la **fragmentation**.

La fragmentation sous-entend que le protocole IP de tout ordinateur IP (système d'extrémité uniquement) doit être en mesure de reconstituer ces fragments pour obtenir le paquet d'origine, par une procédure appelée **réassemblage**.

Chaque fragment d'un paquet IP découpé est un paquet IP complet qui contient un en-tête

IP. Le paquet originel auquel appartient un fragment se reconnait grâce au champ identificateur de l'en-tête IP. Ce champ seul ne suffit pas à déterminer le fragment. On utilise également les champs adresse réseau de l'émetteur, adresse réseau du destinataire et le protocole de la couche supérieure.

Les différents fragments d'un paquet originel peuvent emprunter des itinéraires différents pour parvenir à l'ordinateur cible et être également fragmentés plusieurs fois sur leurs trajets. La position des données d'un fragment au sein du paquet d'origine est identifiée grâce au **décalage de fragment** : les sous-paquets sont repérés par un décalage et non par un numéro parce qu'un sous-paquet peut être fragmenté à tout moment le long de sa route ; on ne peut pas revenir en arrière pour renuméroter les fragments.

Tous les fragments, jusqu'à l'avant-dernier, comportent un bit `MF` (pour l'anglais *More Fragments*) pour indiquer que d'autres fragments vont suivre.

La figure 2.9 ([WPRMB–02], p. 280) montre l'exemple d'un paquet IP qui doit être fragmenté plusieurs fois.

2.6.1.4 Les tribulations d'un paquet IP

Hôte de départ.- Lorsqu'une application doit envoyer un paquet sur le réseau, la partie du sous-système réseau chargée du protocole IPv4 de la couche réseau effectue un certain nombre de tâches simples :

- Elle vérifie tout d'abord si le paquet doit être fragmenté. IPv4 autorise une taille de paquet maximale de 65 535 octets, ce qui est bien supérieur à ce que peuvent traiter la plupart des couches inférieures. Si besoin est, elle crée les fragments.
- Elle construit ensuite, pour chaque fragment, le paquet IP en respectant les longueurs valides, telles qu'elles sont spécifiées par l'implémentation IP locale. Une **somme de contrôle** est calculée pour les données, puis l'en-tête IP est construit.
- Ensuite commence le routage. Si un routage précis est prédéterminé (à titre de contrôle en général), les informations sur celui-ci sont ajoutées à l'en-tête au moyen d'une **option**.
- Enfin le paquet est transmis à la couche inférieure pour être envoyé sur le réseau physique.

Passerelles et routeurs.- À mesure qu'un paquet traverse l'interréseau, chaque passerelle effectue une série de tests :

- Une fois que la couche inférieure a enlevé l'en-tête de la trame reçue, la couche IP de la passerelle calcule la somme de contrôle et vérifie l'intégrité du paquet. Si les sommes de contrôle ne correspondent pas, le paquet est détruit et un message d'erreur est envoyé au composant émetteur *via* ICMP.
- Ensuite, le champ TTL de « durée de vie » de l'en-tête IP est décrémenté puis examiné. Si la durée de vie du paquet a expiré, le paquet est écarté et un message d'erreur est envoyé au composant émetteur *via* ICMP.
- Le prochain saut de la route est déterminé, en analysant l'adresse de destination ou à partir du routage spécifique indiqué dans le champ « option » de l'en-tête IP
- Le paquet est reconstruit avec une nouvelle valeur de TTL et donc une nouvelle somme de contrôle.
- Si une fragmentation est nécessaire, parce que le support physique pour atteindre le saut suivant admet des trames de longueur moindre, le paquet est divisé, et de nouveaux paquets, contenant les informations d'en-tête adéquates, sont créés.
- Enfin, le paquet est envoyé à la couche infrieure.

Hôte de destination.- Lorsque le paquet est finalement reçu par le composant récepteur :

— Le système effectue un calcul de la somme de contrôle et vérifie que celle-ci concorde avec le champ adéquat de l'en-tête. Si ce n'est pas le cas, un message ICMP est envoyé à l'émetteur.

— Il regarde ensuite s'il y a des fragments supplémentaires. L'opération inverse de la fragmentation s'appelle le **réassemblage**. Si d'autres paquets sont nécessaires pour réassembler le paquet originel, le système attend, après avoir lancé un **minuteur de réassemblage**, pour ne pas être bloqué si les paquets suivants mettent trop de temps à arriver.

— Si le composant ne peut réassembler toutes les parties du paquet originel avant que le minuteur n'atteigne 0, la partie du paquet arrivée est détruite et un message d'erreur est envoyé à l'émetteur *via* ICMP.

Une des conséquences de cette étape est qu'un paquet fragmenté a moins de chances d'arriver à bon port qu'un paquet non fragmenté, ce qui explique pourquoi la plupart des applications essaient d'éviter autant que possible la fragmentation.

— Enfin l'en-tête IP est enlevé, le paquet originel est reconstruit (s'il a été fragmenté) et il passe à travers les couches pour arriver à la couche d'application.

2.6.2 En-tête IPv4

Rappelons que l'unité d'information utilisée par IP est appelée **paquet**, ou plus précisément **paquet Internet** ou **paquet IP**. Un paquet est constitué d'un en-tête IP suivi des données. La figure suivante montre l'agencement de l'en-tête IP, telle que définie par [RFC 791] (« *Internet Protocol* », écrite par Postel en 1981) :

<table>
<tr><td>Version
(4 bits)</td><td>Longueur
(4 bits)</td><td>Type
(8 bits)</td><td colspan="4">Longueur du paquet
(16 bits)</td></tr>
<tr><td colspan="3">Identificateur</td><td></td><td>DF</td><td>MF</td><td>Décalage</td></tr>
<tr><td colspan="2">TTL</td><td>Protocole</td><td colspan="4">Somme de contrôle</td></tr>
<tr><td colspan="7">Adresse émetteur</td></tr>
<tr><td colspan="7">Adresse destinataire</td></tr>
<tr><td>Options</td><td colspan="6">Remplissage</td></tr>
</table>

Pour être complet cette RFC a été mise à jour par la [RFC 1349] (*Type of Service in the Internet Protocol Suite*), maintenant obsolète et remplacée par la [RFC 2474] (*Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*), mise à jour par la [RFC 3168] (*The Addition of Explicit Congestion Notification (ECN) to IP*) et la [RFC 3260] (*New Terminology and Clarifications for Diffserv*).

Commentons les champs de cet en-tête un à un :

— Le **numéro de version** est un champ de 4 bits contenant le numéro de version IP que le paquet utilise.

La version la plus utilisée est la version 4 mais quelques systèmes effectuent des tests avec la version 6. Nous décrivons la version 4 dans la suite.

Ce champ est *a priori* nécessaire pour que le logiciel IP récepteur sache décoder le reste de l'en-tête, qui varie à chaque nouvelle version de protocole IP. En fait la distinction se fait par le champ *protocole* de l'en-tête Ethernet au niveau de la couche MAC : `0x800` pour IPv4 et `0x86DD` pour IPv6.

— La **longueur de l'en-tête** (`IHL` pour *Internet Header Length*) est un champ de 4 bits qui indique la longueur totale de l'en-tête IP construit par la machine émettrice. Cette longueur permet de déterminer où commencent les données puisqu'il n'existe pas de marqueur de début des données.

Les spécifications d'IP (ainsi que de la plupart des autres protocoles de la suite TCP/IP) définissent comme unité de longueur le **mot** de 32 bits, soit 4 octets.

À cause du champ `Options`, nous ne pourrions pas connaître la longueur de l'en-tête sans ce champ. L'en-tête IP a une longueur maximale de quinze mots, soit 60 octets. Le plus court en-tête autorisé par IP est celui sans aucune option, soit cinq mots ou 20 octets.

— Le champ suivant, de 8 bits (un octet), spécifiait à l'origine le **type de service** (`ToS` pour *Type Of Service*), qui indiquait aux routeurs comment traiter correctement le paquet.

Les 8 bits de ce champ étaient répartis de la façon suivante :

0	3	4	5	6
Priorité	Délai	Débit	Fiabilité	non utilisé

— Les trois premiers bits, 0 à 2, indiquaient l'**ordre de priorité** (*precedence* en anglais) du paquet, avec des valeurs comprises entre 0 (normal) et 7 (contrôle réseau). Plus cette valeur est grande, plus le paquet est important, ce qui, en théorie du moins, le fera arriver plus vite à destination.

— Les trois bits suivants sont des drapeaux d'un bit qui contrôlent le délai (*Normal/low delay* pour le bit 3), le débit (*Normal/High throughput* pour le bit 4) et la fiabilité (*Normal/High reliability* pour le bit 5) du paquet. Un bit positionné à 1 indique un délai bas, un grand débit et une haute fiabilité respectivement.

— Les deux derniers bits (6 et 7) étaient réservés pour des utilisations futures et devaient être mis à zéro.

Dans la mise à jour [RFC 3168], ces deux bits sont utilisés pour traiter la congestion des paquets (*ECN*).

En pratique, la plupart des implémentations des routeurs TCP/IP ignoraient ce champ et traitaient tous les paquets avec les mêms valeurs de priorité, de délai, de débit et de fiabilité. Il prenait donc la plupart du temps la valeur 0.

La signification de ce champ a été changée dans [RFC 2474] pour devenir `DS` (pour *Differentiation Services*). Il indique maintenant le comportement de transmission utilisé, ce qui n'intéresse que les routeurs. Les bits 0 à 5 constituent le `DSCP` (*Differentiated Services Code Point*) et les deux bits restants 6 et 7 sont encore non utilisés.

[RFC 3168] utilise les deux bits 6 et 7 comme `ECN` (*Explicit Congestion Notification*) pour avertir les points terminaux de la congestion des routeurs, ce qui risque de faire disparaître des paquets. Ces deux bits sont appelés `ECT` (pour *ECN Capable Transport*) et `CE` (pour *Congestion Experienced*).

[RFC 3260] change la terminologie et clarifie les utilisations du `DS`.

— La **longueur du paquet** (*Total Length*) est un champ de 16 bits, soit deux octets, qui indique la longueur totale du paquet, en-tête compris, en octets (et non en mots).

On a donc des paquets d'au plus 65 535 octets.

— L'**identificateur de paquet originel** (*Fragment-ID*) est un champ de 16 bits, soit deux octets, qui contient un identificateur unique créé par le nœud émetteur pour chaque paquet, avant que celui-ci ne soit éventuellement fragmenté. Ce numéro est nécessaire lors du réassemblage des fragments, afin de garantir que les fragments d'un paquet ne soient pas mêlés à ceux d'un autre.

Bien entendu ce numéro « unique » doit être réattribué sur une longue période. Il faut juste s'assurer qu'il ne soit pas donner deux fois sur une période relativement courte.

— Le champ **drapeaux** (*Flags* en anglais) est un champ de trois bits qui contrôle la gestion des paquets lorsqu'une fragmentation est requise. Le premier bit n'est pas utilisé, les deux bits restants permettent de spécifier des drapeaux appelés `DF` (pour *Don't Fragment*, ne pas fragmenter) et `MF` (pour *More Fragments*, encore des fragments) :

  — Si le drapeau `DF` vaut 1, le paquet ne peut en aucun cas être fragmenté.

    Si la couche de logiciel IP ne peut pas envoyer le paquet à une autre machine sans le fragmenter et que ce bit vaut 1, le paquet est détruit et un message d'erreur doit être envoyé au composant émetteur *via* ICMP.

  — Si le drapeau `MF` vaut 1, c'est que le paquet en cours est suivi d'autres paquets (parfois appelés **sous-paquets**) qui devront être réassemblés pour former le paquet originel. Le dernier fragment d'un paquet envoyé doit avoir son drapeau `MF` égal à 0, de manière à ce que le composant récepteur sache qu'il doit cesser d'attendre de nouveaux sous-paquets.

    Comme l'ordre d'arrivée des fragments ne correspond pas forcément à leur ordre d'envoi, le drapeau `MF` est utilisé en conjonction avec le champ suivant pour indiquer à la machine réceptrice à quoi ressemble le paquet originel.

— Si le drapeau `MF` est égal à 1, ce qui indique un paquet fragmenté, le champ `décalage de fragment` (*Fragment Offset* en anglais) de 13 bits contient la position, l'unité étant deux mots ou huit octets ou 64 bits, au sein du paquet complet, du début des données contenues dans le paquet en cours. Ceci permet à IP de réassembler les paquets fragmentés dans leur ordre correct.

  Puisque ce champ comporte 13 bits et que les décalages sont calculés en unités de 8 octets, on retrouve bien la longueur de paquet maximale de 65 535 octets. Puisque ce champ a une taille de 13 bits, seulement 8 192 fragments au maximum peuvent appartenir à un paquet IP originel. Tous les fragments, sauf le dernier, doivent avoir une longueur multiple de 8 octets, l'unité élémentaire de fragmentation.

— Le champ TTL de durée de vie, de taille un octet, est, comme nous l'avons vu, un compteur qui indique le nombre maximal de sauts que le paquet peut traverser avant d'être détruit par le nœud en cours.

  Cette durée de vie permet d'éviter que des paquets IP circulent sans fin sur le réseau et ne saturent celui-ci. Elle est fixée par le nœud émetteur lors de l'assemblage du paquet.

— Le **protocole de transport** est un champ d'un octet qui contient le numéro d'identification du protocole de transport duquel provient ou sera confié le paquet.

  Les numéros sont définis par IANA (*Internet Assigned Numbers Authority*, où ils peuvent être trouvés sur leur site Web). Il existe actuellement environ 50 protocoles affectés d'un numéro de transport pour la partie serveur, codifiés dans [RFC 1700]. Les quatre protocoles les plus importants sont ICMP, qui porte le numéro 1, TCP le numéro 6, UDP le numéro 17 et IGMP le numéro 2.

— La **somme de contrôle** (*header checksum*) ne porte que sur l'en-tête IP (et non sur les données) pour accélérer les traitements.

  La somme de contrôle est égale à l'opposé en complément à un de la somme en complément à un sur 16 bits de tous les mots de 16 bits de l'en-tête IP sauf le champ somme de contrôle.

  Pour vérifier la somme de contrôle, il suffit donc d'ajouter cette somme à la valeur du champ *somme de contrôle*, ce qui devrait donner 0, soit `FFh`.

  Comme le champ `TTL` est décrémenté lors du passage de chaque routeur, la somme de

contrôle doit être recalculée par chaque routeur par lequel passe le paquet.

— Les champs **adresse de l'émetteur** (*Source address*) et **adresse de destination** (*Destination address*) contiennent les adresses IP à 32 bits des composants émetteur et récepteur.

— Le champ **Options** est optionnel et, dans le cas où il existe, est composé de plusieurs codes de longueurs variables. Si plus d'une option est utilisée dans un paquet, les options apparaissent les unes à la suite des autres dans l'en-tête IP.

Toutes les options sont contrôlées par un octet, généralement divisé en trois champs : un *drapeau de copie* sur un bit, une *classe d'options* sur deux bits et un *numéro d'option* sur cinq bits.

— Le **drapeau de copie** est utilisé pour indiquer la manière dont l'option est traitée lorsqu'une fragmentation est nécessaire dans un routeur : lorsque ce bit vaut 0, l'option doit être copiée sur le premier fragment mais pas sur les suivants ; si le bit vaut 1, elle doit être copiée sur tous les fragments.

— Les deux bits de la **classe d'options** permet quatre classes. Pour le moment il n'existe que deux classes : la classe 0 indique que l'option s'applique au contrôle du paquet ou du réseau ; la classe 2 indique que l'option a trait au débogage ou à l'administration.

— Les valeurs actuellement prises en charge pour les classes et numéros d'options figurent dans le tableau suivant :

Classe	Numéro	Description
0	0	Marque la fin de la liste d'options
0	1	Aucune option (utilisée pour le remplissage de caractères)
0	2	Options de sécurité (utilisation militaire seulement)
0	3	Routage tolérant
0	7	Active l'enregistrement du routage (ajoute des champs)
0	9	Routage strict
2	4	Marquage de la date activé (ajoute des champs)

Les options les plus intéressantes sont celles qui activent l'enregistrement du routage (*record route*) et de l'heure (*Internet timestamps*). Elles permettent de tracer le passage d'un paquet au travers de l'interréseau, ce qui peut être utile pour émettre des diagnostics.

Le **routage tolérant** (*loose routing* en anglais) fournit une série d'adresses IP par lesquelles le paquet doit passer, mais il permet de prendre n'importe quelle route pour parvenir à chacune de ces adresses (correspondant à des routeurs en général).

Le **routage strict** n'autorise aucune déviation de la route spécifiée. Si cette route ne peut pas être suivie, le paquet est détruit et un message d'erreur est envoyé *via* ICMP. Le routage strict est fréquemment utilisé pour tester les routes mais rarement pour transmettre des paquets utilisateur.

— Le contenu de la **zone de remplissage** (*padding*) dépend des options sélectionnées.

On utilise le remplissage pour s'assurer que la longueur de l'en-tête est bien un multiple de mots, c'est-à-dire de 32 bits.

2.6.3 Analyse d'un en-tête IP par Wireshark

Si nous revenons à notre capture, nous nous apercevons que la troisième trame comprend un en-tête IP, plus exactement un en-tête Ethernet, un en-tête IP, un en-tête UDP et un message DNS (voir figure 2.10). Rappelons que c'est l'en-tête Ethernet qui nous indique, grâce à son champ `type`, que l'en-tête Ethernet est suivie d'un en-tête IP. La longueur totale de celui-ci sera indiquée dans l'en-tête lui-même.

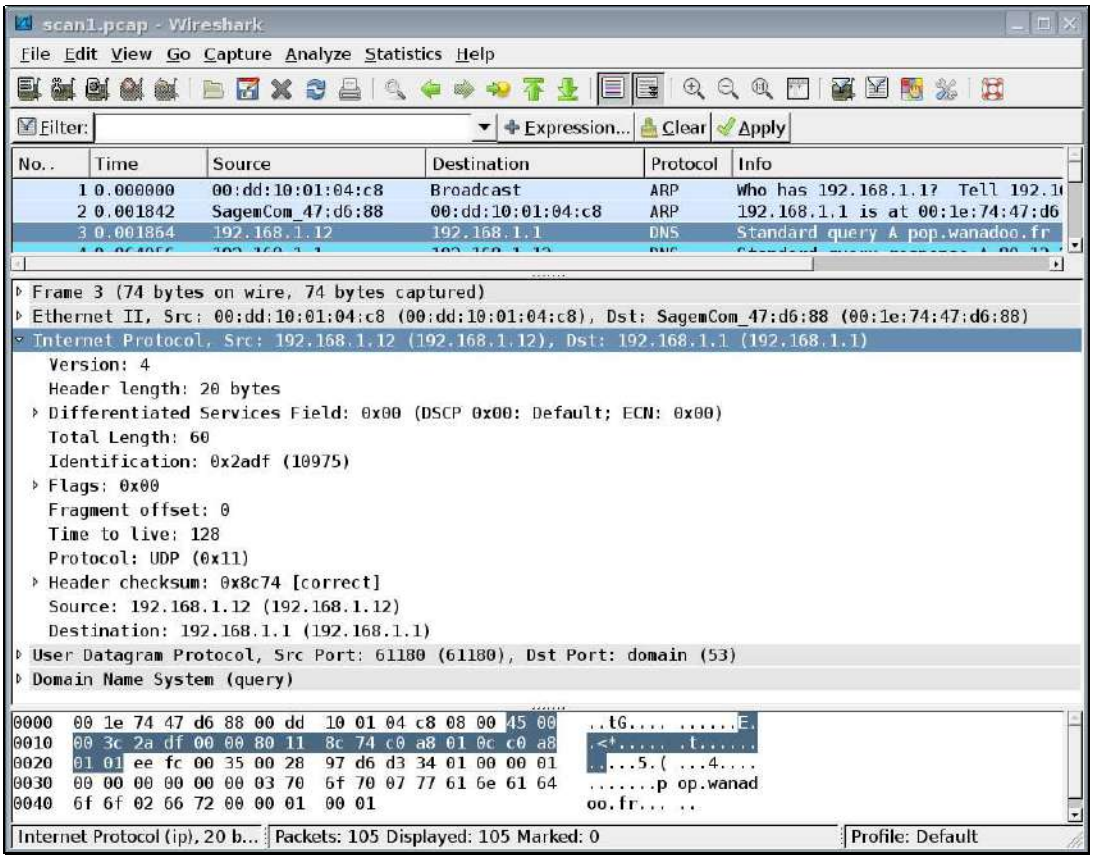

FIGURE 2.10 – Analyse d'un en-tête IP

— Les quatre premiers bits (le premier demi-octet) confirment qu'il s'agit de la version quatre d'IP et donc d'un en-tête IPv4.

— Les quatre bits suivants (le demi-octet suivant) indiquent que la longueur de l'en-tête est de cinq mots, soit vingt octets. Il s'agit ici de la longueur minimum d'un en-tête : il n'y a donc pas d'option.

— L'octet suivant, interprété comme *Differentiated Services Field* par `Wireshark`, est nul, ce qui sera le plus souvent le cas pour une extrémité.

— Les deux octets suivants précisent la longueur du paquet, ici `38h`, soit 60 octets. Avec les 14 octets de l'en-tête Ethernet, on retrouve bien les 74 octets de la trame.

— L'identificateur de paquet de deux octets (déterminé de façon presque aléatoire, rappelons-le) est ici `2ADFh`, soit 10 975 (mais la conversion en décimal est sans intérêt).

— Les trois bits suivants `000b` constituent les drapeaux.
  - Le premier bit est un bit réservé pour une utilisation future, qui est nul.
    Lors de l'envoi, on a intérêt à le positionner à 0. Lors de la réception, il ne faut pas rejeter le paquet si ce drapeau n'est pas nul car il faut conserver le cas d'une évolution.
  - Le bit suivant, `DF`, positionné à 0, indique que l'on peut fragmenter le paquet.
  - Le bit suivant, `MF`, positionné à 0, indique qu'il n'y a pas de paquet suivant portant le même identificateur. Il s'agit donc du dernier fragment s'il y a eu fragmentation.

— Les treize bits suivants, tous égaux à 0, spécifient la position au sein du message complet. Nous en sommes donc au début et, puisque `MF` est égal à zéro, le paquet contient le message complet : il n'y a pas eu de fragmentation.

— L'octet suivant, de valeur `80h` ou 128, indique la durée de vie (`TTL`) du paquet.

— L'octet suivant spécifie le protocole de transport. On a ici `11h`, soit 17. Le protocole de transport est donc UDP.

— Les deux octets suivants constituent la somme de contrôle. On a ici `8C74h`. `Wireshark` indique qu'elle est correcte. Je vous laisse le vérifier.

— Les quatre octets suivants donnent l'adresse IP de l'émetteur. On a ici `C0A8010Ch`, soit 192.168.1.12. On retouve bien l'adresse IP de mon ordinateur.

— Les quatre octets suivants donnent l'adresse IP du destinataire. On a ici `C0A80101h`, soit 192.168.1.1. On retouve bien l'adresse IP de mon routeur.

2.7 Le protocole de couche de transport UDP

Le protocole de la couche de transport UDP (*User Datagram Protocol*), décrit dans [RFC 768], a pour but principal le multiplexage/démultiplexage des applications.

2.7.1 Multiplexage/démultiplexage des applications

Le protocole UDP possède la fonctionnalité essentielle des protocoles de la couche de transport, celle du multiplexage/démultiplexage des applications, c'est-à-dire de transmettre les données des datagrammes à l'application adéquate d'un système terminal, grâce à la spécification des numéros de port.

Notion de port.- Un ordinateur peut communiquer en même temps avec plusieurs autres (en utilisant des processus différents et du pseudo-parallélisme), et même à propos de plusieurs applications d'un même ordinateur : par exemple recevoir une page web et du courrier tout en envoyant un fichier par `ftp`. Les paquets qui arrivent, au niveau de la couche réseau donc, sont triés :

- ceux qui ne correspondent pas à l'adresse IP de l'ordinateur ne lui sont pas communiqués, et ceci en fait dès le traitement des en-têtes de trames ;
- les messages des autres, donc après avoir enlevé les en-têtes de la couche réseau et de la couche transport, sont envoyés vers telle ou telle (instance d') application.

Comment déterminer l'instance d'application vers laquelle on doit envoyer le message ? L'en-tête de la couche de transport (que ce soit TCP ou UDP) contient un champ indiquant celle-ci. Il s'agit d'une entier de 16 bits, donc compris entre 0 et 65 535, appelé **port**.

Modèle client/serveur.- La manipulation des datagrammes se fait toujours dans le cadre du modèle client/serveur :

- le **client** est celui qui demande quelque chose ; ce quelque chose est appelé une **requête** ;
- le **serveur** est celui qui attend que l'on demanque quelque chose. On appelle **réponse** ce qu'il renvoie au client. Il faut donc prévoir pour le serveur un procédé pour cette attente, ne serait-ce qu'une simple boucle.

Attribution des numéros de port aux serveurs.- Si un client situé sur un ordinateur A veut envoyer une requête à un serveur situé sur un ordinateur B, comment peut-il connaître son port ? Chaque **service** peut-il choisir son port de façon arbitraire ? L'attribution des ports est-elle centralisée ? Comment une application peut-elle obtenir le numéro de port d'une autre application ?

Un organisme dont nous avons parlé, à savoir **IANA** (pour *Internet Assigned Number Authority*, rappelons-le), attribue des ports fixes aux serveurs des applications connues. On parle alors des **ports bien connus** (*well bnown ports*), auxquels chaque application peut se référer lors d'une requête. Au départ, les valeurs situées entre 0 et 256 étaient réservées à l'attribution des ports normalisés. Mais, en 1994, ce quota a été étendu aux valeurs contenues entre 0 et 1 023, en raison de la demande toujours croissante de ports réservés. La liste des ports normalisés peut être consultée dans [RFC 1700] de 1994, et nos jours sur le site Web de IANA :

`http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`

La figure 2.11 montre quelques-uns des numéros de ports les plus utilisés par les serveurs.

Attribution dynamique pour les clients.- Si les services bien connus ont des numéros fixés par IANA, les clients par contre se voient toujours attribuer un numéro de port de façon dynamique

Application	Port
daytime (heure du jour)	13
ftp	21
telnet	23
smtp	25
time (heure)	37
DNS	53
finger	79
web	80
pop3	110
nntp (news)	119
snmp	161
irc (Internet Relay Chat)	194

FIGURE 2.11 – Numéros de port bien connus des serveurs

par le sous-système réseau du système d'exploitation au moment où ils ont besoin (supérieurs évidemment à 1 024).

2.7.2 Utilisation d'UDP

Le protocole UDP est utilisé, d'une part, pour les applications orientées **transaction** telles que DNS, dans lesquelles seules une requête et une réponse associée doivent être transmise, de telle sorte qu'il est inutile d'établir une connexion pour cela. D'autre part, le protocole UDP est également utilisé dans les contextes où la rapidité de la transmission des données prédomine sur la fiabilité. C'est ainsi que dans le cas des flux audio, qui sont transmis sous forme de petits messages, il n'est pas excessivement gênant que quelques messages isolés se perdent ; par contre, un contrôle automatique de flux et des erreurs (assorti de la retransmission des messages perdus) s'avérerait souvent très pénalisant pour la régularité du flux.

2.7.3 L'en-tête UDP

L'en-tête UDP est défini dans [RFC 768]. Sa structure, très simple, n'occupe que huit octets, comme le montre la figure suivante :

Port source (16 bits)	Port Destination (16 bits)
Longueur (16 bits)	Somme de contrôle (16 bits)

— Le **port source** (*source port*) est un champ, dont le renseignement est facultatif, contenant le numéro de port de l'expéditeur, compris entre 1 et 65 535. Si aucun numéro de port n'est spécifié, le champ est mis à 0. Ce champ est par contre nécessaire au destinataire s'il doit renvoyer des données.

— Le **port de destination** (*destination port*) est le numéro de port sur la machine de destination.

— La **longueur** (*length*) est la taille du datagramme, exprimée en nombre d'octets, comprenant en-tête et données. Sa valeur minimum est 8 (taille de l'en-tête UDP) alors que le datagramme UDP le plus long peut transporter 65 535 - 8 = 65 527 octets de données utiles.

— la **somme de contrôle** (*checksum*) est un champ facultatif sur lequel nous allons revenir ci-dessous. De nombreuses applications n'y ont pas recours. S'il n'est pas utilisé, sa valeur doit être zéro.

L'en-tête UDP est suivi des *données*. Elles sont de longueur variable. Un **remplissage** éventuel assure que le datagramme a une longueur multiple de 16 bits.

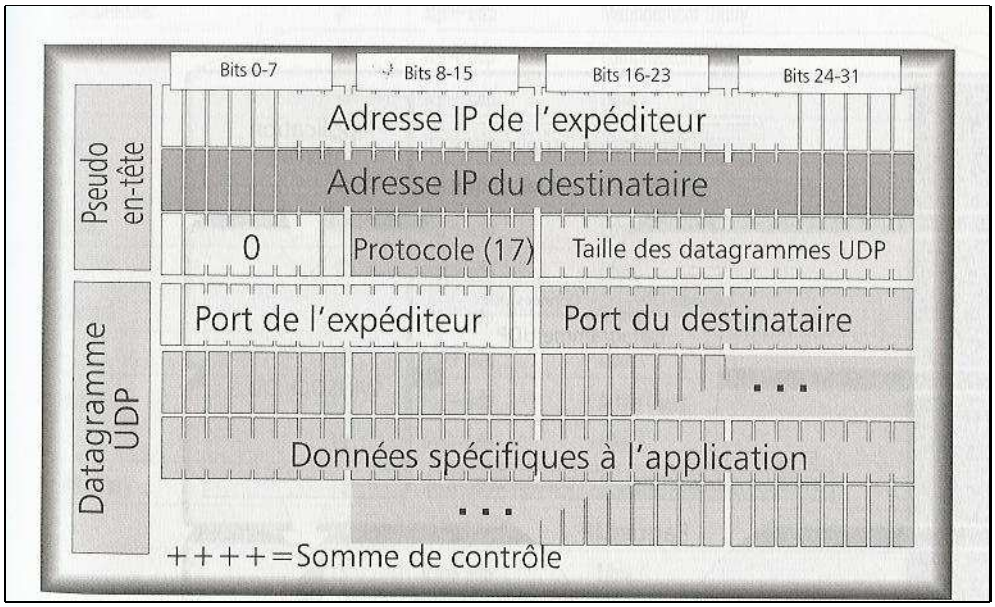

FIGURE 2.12 – Pseudo en-tête UDP

2.7.4 Sécurisation optionnelle par somme de contrôle

UDP permet une **connexion non fiable** dans le sens où il n'existe aucun mécanisme permettant de reconnaître et de traiter les datagrammes perdus ou dupliqués. Les datagrammes peuvent cependant être protégés, de manière optionnelle, des erreurs à l'aide d'une somme de contrôle.

Contrairement au protocole IP, cette somme de contrôle ne se rapporte pas uniquement à l'en-tête du datagramme mais également aux données utiles de celui-ci. Si le datagramme expédié comporte une somme de contrôle, celle-ci est le complément à 1 de 16 bits du complément à 1 de la somme des bi-octets constituant le datagramme (en-tête et données utiles) ainsi que d'un **pseudo en-tête**, dont la structure est montrée à la figure 2-12 ([TJ–97], p.117) : il contient les adresses IP de l'expéditeur et du destinataire, la taille du datagramme UDP et le code de protocole IP pour UDP (à savoir 17) ; toutes ces informations proviennent de l'en-tête de la couche réseau.

Lorsque le calcul de la somme de contrôle donne la valeur nulle, on indique `0xFFFF`, soit -1, à la place, qui équivaut également à 0 sans toutefois coïncider avec la valeur constante 0 signalant l'absence de somme de contrôle.

La méthode de calcul de la somme de contrôle s'implémente très efficacement pour tous les types de processeurs. Elle est décrite dans les recommandations [RFC 1071], [RFC 1141] et [RFC 1624].

2.7.5 Analyse d'un en-tête UDP

Continuons à analyser la trame DNS (voir figure 2.13). L'en-tête IP nous a indiqué, grâce à son champ `protocole de transport` que la suite est un en-tête UDP. La longueur de celui-ci est toujours de huit octets.

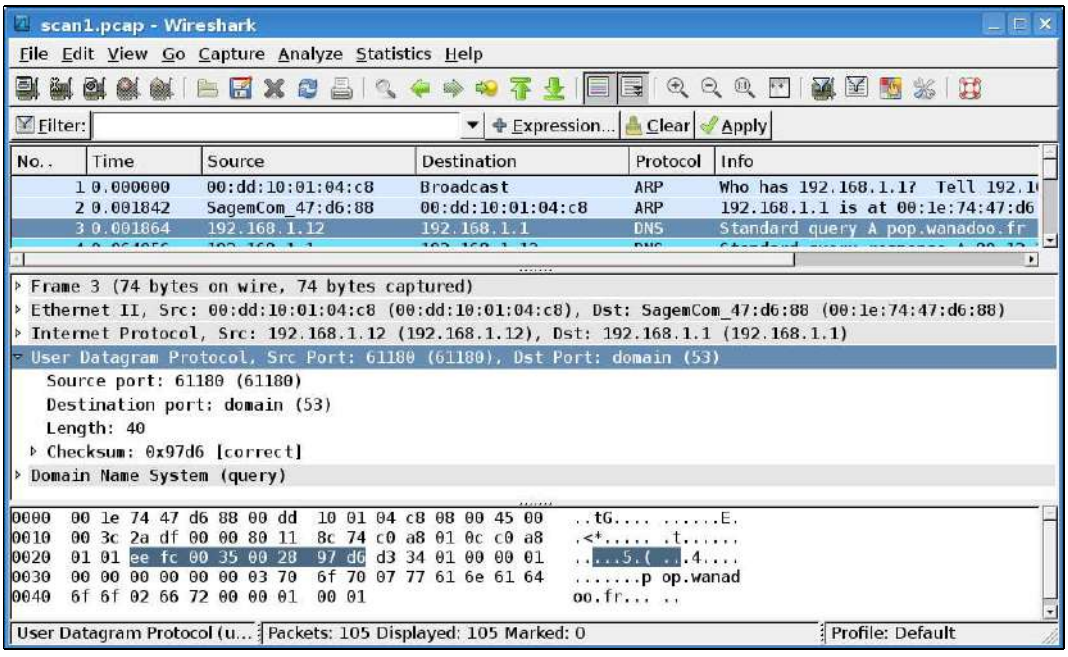

FIGURE 2.13 – Analyse d'un en-tête UDP

— Les deux premiers octets de l'en-tête UDP spécifient le numéro de port source. On a ici `EEFCh`, soit 61 180. Rappelons que ce numéro de port est déterminé (presqu'au hasard) par l'implémentation de la couche UDP du sous-système réseau du système d'exploitation de façon à ce que deux instances d'applications différentes n'aient pas le même numéro de port.

— Les deux octets suivants spécifient le numéro de port destination. On a ici `0035h`, soit 53. Il s'agit d'un numéro de port bien connu, celui attribué pour le serveur DNS. Ceci nous indique que le message concerne un message DNS, et même une requête puisque sinon le port serait supérieur à 1 024.

— Les deux octets suivants indiquent la taille du datagramme. Elle est ici de `00 28h`, soit 40 octets. Avec les vingt octets de l'en-tête IP et les quatorze octets de l'en-tête Ethernet, on retrouve bien les 74 octets de la trame.

— Les deux derniers octets constituent la somme de contrôle. On a ici `97 D6h`. `Wireshark` indique qu'elle est correcte ; nous vous laissons le soin de le vérifier.

2.8 Le protocole de couche de transport TCP

2.8.1 Fonctionnalités

Le protocole de transport TCP permet, comme UDP, le multiplexage/démultiplexage des applications mais il essaie d'aller au-delà en s'intéressant à la fiabilité (en transférant des données à nouveau si celles-ci ont été altérées en cours de route), au contrôle du flux (en réduisant celui-ci en cas de congestion ou, au contraire, en l'augmentant progressivement sinon) et à la connexion (ce qui évite de renvoyer les mêmes informations).

L'octet comme granularité.- Une application constitue un *message* qui doit être envoyé et reçu, de taille arbitraire. TCP attribue (virtuellement) un numéro à chaque octet du message qu'il envoie : on appelle cela le **séquencement des octets**. Les octets sont remis à l'application à l'autre extrémité dans leur ordre d'envoi.

Le séquencement des octets n'empêche pas TCP d'envoyer le message par groupe d'octets, bien qu'il contrôle par octet. Pour cela les connexions TCP négocient généralement une taille maximum de segment lors de la connexion. TCP attend alors que la taille maximum négociée soit atteinte avant d'envoyer un segment. Le message n'a aucune raison d'avoir une taille multiple de la taille maximum négociée. Une fonction `push()` est donc nécessaire pour envoyer la fin du message.

Fiabilité.- La remise de façon fiable, c'est-à-dire à coup sûr, du flux d'octets est impossible. Cependant TCP possède un mécanisme qui essaie de faire au mieux. Le mécanisme **PAR** (*Positive Acknowledgment Retransmission*, retransmission d'acquittement positif) affecte un **numéro de séquence** à chaque segment transmis et requiert un accusé de réception (**ACK**) de la part du module TCP récepteur ; si cet accusé de réception ne parvient pas avant expiration d'un certain délai, le module TCP de l'expéditeur doit retransmettre le segment concerné.

Au niveau du module TCP récepteur les numéros de séquence servent à ordonner les segments et à éliminer les doublons.

Pour détecter les données erronées, on se sert d'un champ « somme de contrôle ». Les segments dont la somme de contrôle ne correspond pas sont purement et simplement écartés.

Contrôle de flux.- Les machines qui émettent et reçoivent les segments TCP ne le font pas toutes au même rythme en raison de la diversité des unités centrales et de la bande passante. Il peut donc arriver que l'émetteur envoie ses données beaucoup plus rapidement que le récepteur ne peut les gérer. C'est pourquoi TCP implémente un mécanisme de contrôle des flux de données.

TCP se sert de la technique dite de la **fenêtre glissante** (voir figure 2.14). Lors de l'ouverture de la connexion, le module TCP du destinataire envoie à l'expéditeur un segment contenant un champ appelé **taille de la fenêtre** : il indique à l'expéditeur le nombre d'octets que celui-ci peut transmettre sans avoir besoin d'une autorisation de la part du destinataire. Voici alors comment fonctionne ensuite le mécanisme de contrôle de flux :

- Les octets qui précèdent la limite inférieure de la fenêtre ont été envoyés par le module expéditeur, qui a reçu un accusé de réception de la part du module TCP du destinataire.
- Les octets se trouvant dans l'intervalle défini par la taille de la fenêtre peuvent être envoyés. S'ils ont déjà été envoyés, leur accusé de réception n'est pas encore parvenu à l'expéditeur.
- Les octets qui dépassent la limite supérieure de la fenêtre n'ont pas encore été envoyés et ne pourront l'être que lorsqu'ils passeront dans l'intervalle de la fenêtre.

La limite inférieure de la fenêtre est le plus petit numéro d'octet n'ayant pas reçu d'accusé de réception. La fenêtre progresse, on dit de gauche à droite, à chaque fois que l'expéditeur reçoit

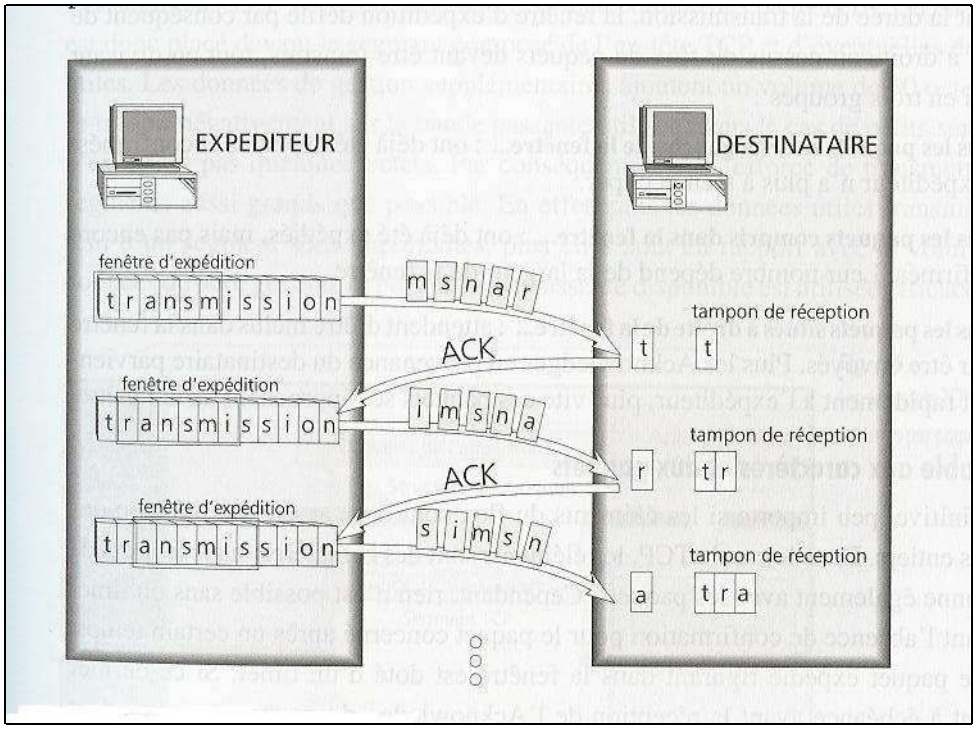

FIGURE 2.14 – Fenêtre glissante

un accusé de réception pour les données expédiées.

La taille de la fenêtre dépend de la quantité d'espace libre encore disponible dans le tampon de données du destinataire. Lorsque le destinataire est trop sollicité, il envoie une taille de fenêtre réduite. Dans les cas extrêmes, cette taille peut se réduire à 1. Lorsque le module TCP du destinataire renvoie une taille de fenêtre égale à zéro, cela indique à l'expéditeur que ses tampons sont pleins et qu'il ne faut plus lui envoyer aucune donnée.

Connexion.- Avant de pouvoir envoyer des données à l'aide de TCP, les processus doivent d'abord établir une *connexion*. Une connexion s'établit entre le numéro de port de l'expéditeur et celui du destinataire. Une **extrémité** (*peer* en anglais) est définie par un couple : l'adresse IP et le numéro de port. On établit une **connexion** entre deux extrémités, autrement dit avec un quadruplet :

(adresse IP 1, numéro de port 1, adresse IP 2, numéro de port 2)

2.8.2 Structure d'un en-tête TCP

La figure suivante montre l'agencement de l'en-tête TCP telle que définie par la [RFC 793] (« *Transmission Control Protocol* ») écrite par POSTEL en 1981 :

Port source (16 bits)								Port destination (16 bits)
Numéro de séquence (32 bits)								
Numéro d'accusé de réception (32 bits)								
Décalage des données (4 bits)	Réservé (6 bits)	URG	ACK	PSH	RST	SYN	FIN	Fenêtre (16 bits)
Somme de contrôle (16 bits)								Pointeur Urgent (16 bits)
Options								Remplissage

— Le `port source` (*Source Port*) est un champ de 16 bits qui identifie l'expéditeur TCP, comme dans le cas d'UDP.

— Le `port de destination` (*Destination Port*) est un champ de 16 bits qui identifie le destinatiare TCP.

— Le `numéro de séquence` (*Sequence Number*) est un champ de 32 bits qui indique la position du bloc de données en cours dans l'ensemble du message.

Pour le premier segment TCP, on indique 0 mais pas toujours. Si l'on transmet 300 octets dans ce segment, le segment suivant débutera à l'octet 301 du message et son champ `numéro de séquence` indiquera 301.

Si le drapeau SYN (voir ci-dessous) vaut 1, ce champ définit le numéro de séquence initial (**ISN**) correspondant à la connexion en cours, ce qui est important si on ne commence pas par 0.

— Le `numéro d'accusé de réception` (*Acknowledgement Number*) est un champ de 32 bits. L'expéditeur utilise cette information pour préciser à son correspondant le numéro de séquence qu'il attend dans le segment TCP suivant. Ceci donne une idée des confirmations (accusés de réception) que l'expéditeur a déjà recues.

Le destinataire doit tenir compte de ce champ lorsqu'il reçoit un en-tête TCP contenant le drapeau `ACK` (voir ci-dessous). Ce drapeau indique que le segment contient, outre les données éventuellement présentes, un accusé de réception.

Pendant qu'il sert au transport des données du point A vers le point B, le segment signale à B les données déjà reçues par A. Bien que circulant de A à B au sein du segment, cette partie du segment concerne donc la communication entre B et A. Cette astuce permet l'envoi d'un accusé de réception sans avoir à envoyer un segment TCP comprenant uniquement un en-tête TCP, comme le montre la figure 2.15 ([TJ–97], p.146). En anglais ce procédé est appelé *piggybacking*.

Un accusé de réception peut servir à acquitter plusieurs segments TCP.

L'accusé de réception signifie que le module TCP a reçu les données, mais il ne garantit pas que les données sont parvenues à l'application.

— Le `décalage de données` (*Data Offset*) est un champ de quatre bits qui indique à quel mot (de 32 bits) commencent les données TCP, qui suivent l'en-tête TCP.

Il pourrait aussi s'appeler longueur de l'en-tête (*Header Length*) car il contient le nombre de mots constituant l'en-tête TCP.

Cette information est nécessaire parce que, comme dans le cas de IP, le champ « options » a une longueur variable. Lorsque le champ « options » est vide, la valeur de décalage des données est cinq (soit 20 octets).

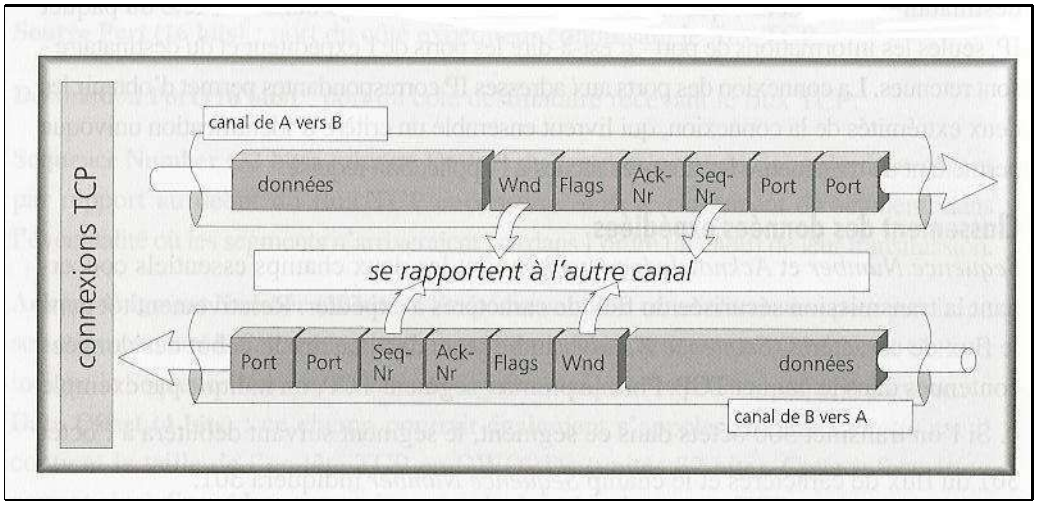

FIGURE 2.15 – Piggybacking

— `Réservé` est un champ de 6 bits réservé pour des utilisations ultérieures.

L'avant dernier de ces bits est devenu le drapeau `CWR` (pour *Congestion Window Reduced*) avec la [RFC 3268]. Il est utilisé par l'expéditeur pour informer le destinataire, s'il est activé (c'est-à-dire de valeur 1), que la fenêtre a été réduite. On doit alors envoyer moins de données par unité de temps.

Le dernier de ces bits est devenu le drapeau `ECE` (pour *ECN Echo*) avec la [RFC 3268]. Il est utilisé par le destinataire pour confirmer à l'expéditeur, s'il est activé, qu'il a reçu un drapeau `CWR`.

— La drapeau `Urg` (pour *URGent*), s'il est activé (valeur égale à 1), indique que le segment contient des données urgentes, c'est-à-dire qu'il faut prendre en compte la valeur du champ de pointeur urgent.

— Le drapeau `Ack` (pour *ACKnowledgment*), s'il est activé, indique que le segment contient un accusé de réception, c'est-à-dire qu'il faut prendre en compte la valeur du champ `Numéro d'accusé de réception`.

— Le drapeau `Psh` (pour *PuSH*), s'il est activé, indique qu'il faut faire suivre immédiatement les données reçues (par exemple en faisant appel à la fonction `push()` sous UNIX), sans attendre que le tampon TCP soit plein.

— Le drapeau `Rst` (pour *ReSeT*), s'il est activé, indique que la connexion doit être réinitialisée pour cause d'erreurs irrécupérables. À la réception du drapeau `Rst`, le destinataire doit immédiatement terminer la connexion.

— Le drapeau `Syn` (pour *SYNchronize sequence numbers*), s'il est activé, indique que les numéros de séquence doivent être synchronisés. Ce drapeau est utilisé lors de l'établissement d'une connexion, par le segment qui ouvre la connexion et par le segment SYN/ACK qui lui répond. Il ne doit pas être utilisé par d'autres segments.

— Le drapeau `Fin`, s'il est activé, indique que l'émetteur n'a plus de données à envoyer. C'est l'équivalent d'un marqueur de fin de transmission. Lorsque l'autre extrémité reçoit un segment avec ce drapeau activé, elle doit répondre par FIN/ACK.

Une fois qu'une extrémité a envoyé un segment contenant ce drapeau, elle ne doit plus envoyer aucun segment. Cependant, l'autre extrémité peut continuer à envoyer des données jusqu'à ce qu'elle ait terminé et envoyer alors un segment FIN et attendre le

FIN/ACK final. Après cela, la connexion est close.

— La `taille de la fenêtre` (*window*) est un champ de 16 bits dont nous avons expliqué ci-dessus la signification.

— La `somme de contrôle` (*checksum*) est calculée, comme pour IP, en prenant le complément à un en 16 bits du complément à un de la somme des mots de 16 bits mais, contrairement à IP, pour le pseudo en-tête et le texte réunis.

Le **pseudo en-tête** de 96 bits contient l'adresse source, l'adresse de destination, l'identificateur de protocole et la longueur du segment.

— On ne doit tenir compte du champ `pointeur urgent` (*Urgent Pointer*) que si le drapeau `Urg` est activé. Il indique la portion des données du message qui est urgente, en en spécifiant le décalage par rapport au numéro de séquence.

Aucune action spécifique n'est entreprise par TCP en ce qui concerne les données urgentes. C'est l'application qui s'en charge éventuellement.

— `Options` : chaque option consiste en un type d'option (1 octet), suivi éventuellement du nombre d'octets dans l'option (1 octet) puis des données (de taille le nombre d'octets précédent moins deux). Dans la RFC 793, seules trois types sont définis pour TCP :

  — 0 : fin de la liste d'option (*End-of-Options*), option constituée d'un seul octet.

  L'utilisation de cette option n'est pas nécessaire lorsque la fin des options coïncide avec la fin de l'en-tête TCP.

  — 1 : pas d'opération (*No-Operation*), option constituée d'un seul octet. Ce type sert à aligner le commencement de l'option suivante sur le début d'un mot, mais les expéditeurs TCP ne prennent pas toujours cette précaution.

  — 2 : taille de segment maximale (`MMS` pour *Maximum Segment Size*) de longueur 4. Ce champ sert à spécifier la taille maximale de tampon qu'une implémentation TCP destinataire peut accepter.

— Le champ `remplissage` (*padding*) est rempli pour être sûr que la longueur de l'en-tête est un multiple de 32 bits.

2.8.3 Déroulement d'une connexion

Négociation d'ouverture d'une session TCP en trois étapes.- On ouvre une session TCP à l'aide d'une procédure de négociation en trois temps (*Three-way handshake* en anglais) :

- Le client envoie au serveur un segment TCP d'*ouverture de session* caractérisé par les valeurs suivantes des drapeaux : SYN = 1 et ACK = 0. Le numéro de séquence d'ouverture est SEQ = X, appelé **ISS** pour *Initial Sender Sequence*. On a éventuellement Z octets de données mais en général Z = 0. Le numéro d'accusé de réception est nul : #ACK = 0.
- Le serveur, s'il accepte d'établir une connexion avec le client, accuse réception de ce segment d'ouverture par un segment TCP caractérisé par les valeurs suivantes des drapeaux : SYN = 1 et ACK = 1. Le numéro de séquence d'ouverture de la part du serveur est SEQ = Y, appelé **IRS** pour *Initial Receiver Sequence*. Le numéro d'acquittement doit être #ACK = X + 1 + Z.

  Si le serveur n'accepte pas d'établir une connexion avec le client, il envoie un segment au client avec les drapeaux SYN et RST positionnés.
- Si le serveur a accepté d'établir une connexion alors, dans l'étape trois, le client accuse réception du numéro de séquence que lui a envoyé le serveur avec les valeurs suivantes des drapeaux : SYN = 0 et ACK = 1. On doit avoir SEQ = X + 1 + Z et #ACK = Y + 1.

Transmission des données.- À ce moment la connexion a été établie, les données peuvent être envoyées.

Après l'établissement de la connexion TCP, le drapeau ACK des segments doit être positionné pour indiquer que le champ « numéro d'accusé de réception » est valide. La transmission des données peut se faire dans les deux sens, jusqu'à la fermeture de la session.

Fin de session.- Pour clore la session, une des deux extrémités envoie le drapeau FIN. Mais la fermeture de la connexion TCP n'est réalisée que lorsque l'autre extrémité répond par les drapeaux FIN/ACK pour accepter la fermeture. Cette façon de faire permet d'éviter les pertes de données que pourrait occasionner une fermeture unilatérale.

2.8.4 Analyse d'en-têtes TCP

Pour utiliser le protocole de transport TCP, chargeons par exemple la page d'accueil de notre université[1] dans un navigateur, après avoir fait démarrer une capture `Wireshark` :

`http://www.univ-paris12.fr`

et arrêtons la capture. On récupère plus de de cinq cents trames, mais nous n'allons en analyser que quelques-unes.

2.8.4.1 Première trame TCP (SYN)

Les deux premières trames concernent DNS, utilisant UDP comme nous l'avons déjà vu. Considérons la première trame TCP de cette capture (voir figure 2.16). Il n'y a pas grand chose à dire sur les en-têtes Ethernet et IP, sinon que le protocole de transport de ce dernier en-tête est `06h`, soit 6, pour indiquer TCP. Passons à l'en-tête TCP.

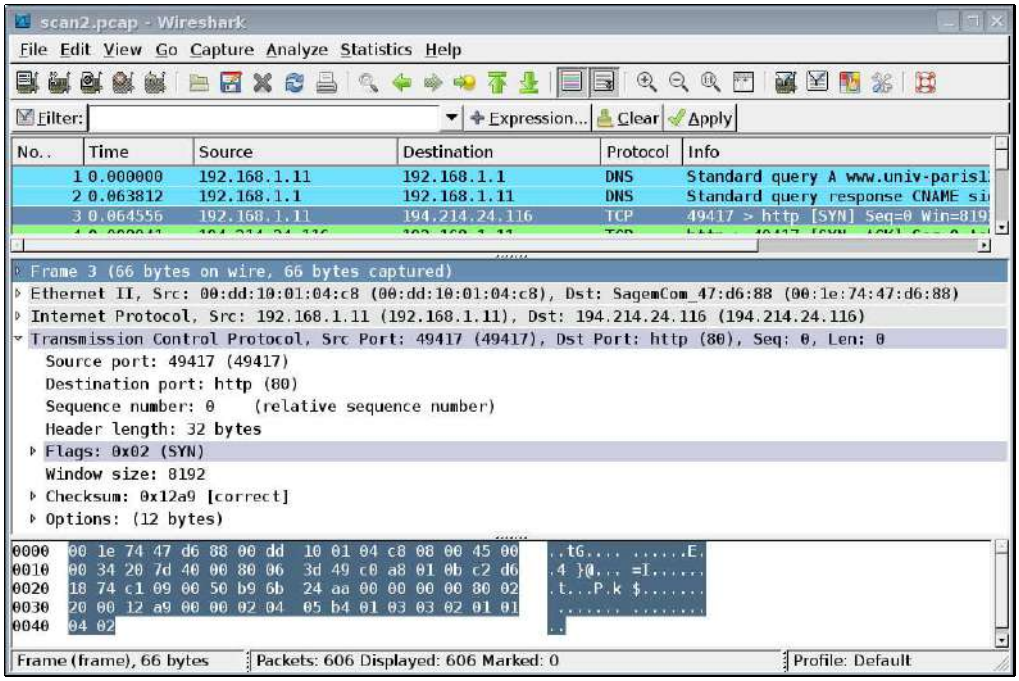

FIGURE 2.16 – Analyse d'un en-tête TCP (SYN)

— Les deux premiers octets de l'en-tête TCP spécifient le numéro de port source. On a ici `C1 09h`, soit 49 417. Rappelons que, comme pour UDP, ce numéro de port est déterminé (presque au hasard) par le module TCP ,de façon à ce que deux applications différentes n'aient pas le même numéro de port pour la partie client, tout au moins sur une durée raisonnable.

— Les deux octets suivants spécifient le numéro de port destination. On a ici `00 50h`, soit 80. Il s'agit d'un numéro de port bien connu, celui attribué aux services HTTP. Ceci nous indique que le message concerne *a priori* une requête HTTP mais, en fait, comme

1. Le nom de cette page d'accueil est toujours valable, mais s'affiche `www.u-pec.fr`.

nous le verrons, il n'y aura pas de message HTTP : il s'agit en fait de la première trame d'ouverture d'une session TCP.

— Les quatre octets suivants indiquent le numéro de séquence. On a ici `B9 6B 24 AAh`. Il n'est donc pas nul. Remarquons que l'analyseur lui attribue le numéro *relatif* 0.

— Les quatre octets suivants indiquent le numéro d'accusé de réception. On a ici zéro, ce qui est normal puisque le client essaie d'initier une connexion : il n' a donc encore reçu aucun octet du serveur. De toute façon, on n'a pas à tenir compte de ce nombre puisque, comme nous le verrons ci-dessous, le drapeau `Ack` n'est pas positionné.

— Les quatre bits suivants indiquent la longueur de l'en-tête TCP en mots de 32 bits. On a ici `8h`, soit 32 octets : on a donc le minimum de 20 octets et 12 octets d'options.

— Les six bits suivants, tous à zéro, sont réservés pour une utilisation ultérieure.

— Les six bits suivants indiquent la valeur des drapeaux. On a `02h`, soit `000010b` : les drapeaux `Urg`, `Ack`, `Psh`, `Rst` et `Fin` ne sont pas positionnés. Le drapeau `Syn` est positionné : il indique le début d'une connexion TCP.

— Les deux octets suivants indiquent la longueur de la fenêtre. On a ici `20 D0h`, soit 8 192 octets.

— Les deux octets suivants donnent la valeur de la somme de contrôle. On a ici `12 A9h` et l'analyseur dit qu'elle est correcte. Nous vous laissons le soin de le vérifier.

— Les douze octets suivants concernent les options. Ne nous intéressons ici qu'à la première des options spécifiées. Elle est du type 2, donc spécifie la taille maximum du segment, qui est de `05 B4h`, soit 1 460 octets.

2.8.4.2 Deuxième trame TCP (SYN/ACK)

On peut analyser de la même façon la deuxième de ces trames (figure 2.17). Ne commentons que ce qui est intéressant :

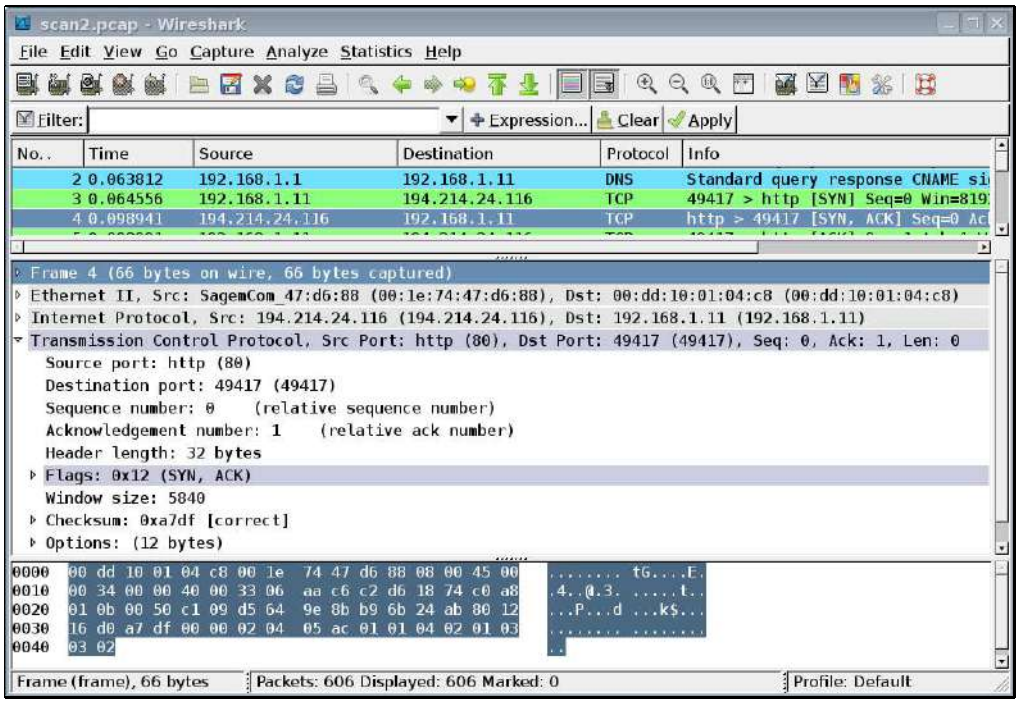

FIGURE 2.17 – Analyse d'un en-tête TCP (SYN/ACK)

— Le numéro de séquence est ici `D5 64 9E 8Bh`, différent donc de celui de la trame précédente : il n'y a aucune raison que le client et le serveur donnent des numéros de séquence identiques.

— Le numéro d'accusé de réception est `B9 6B 24 ABh`, soit le numéro de séquence du segment précédent plus un. Le serveur a donc bien reçu tous les octets du segment envoyé par le client.

— La valeur de la suite de drapeaux est `12h`, soit `010010b` : les drapeaux `Urg`, `Psh`, `Rst` et `Fin` ne sont pas positionnés alors que les drapeaux `Ack` et `Syn` le sont. Ils indiquent que le numéro d'accusé de réception doit être pris en compte et la demande d'ouverture d'une connexion connexion TCP.

— La longueur de la fenêtre est `16 D0h`, soit 5 840, qui est bien inférieure à celle du client (8 192 octets).

2.8.4.3 Troisième trame TCP (ACK)

Ne commentons que ce qui est intéressant pour la troisième trame TCP (figure 2.18) :

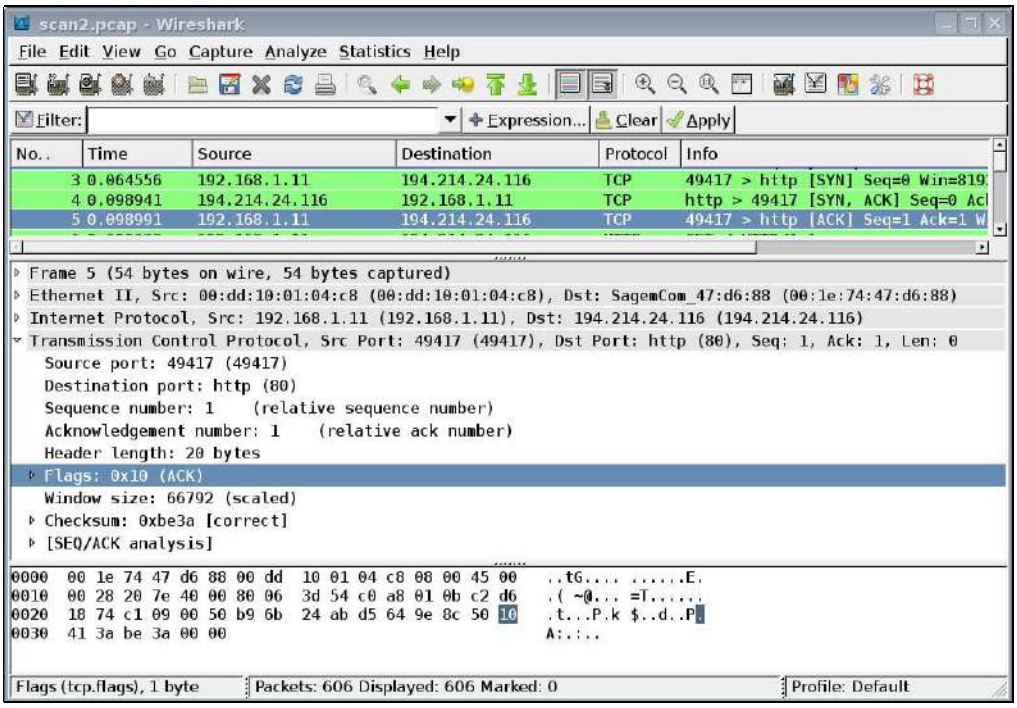

FIGURE 2.18 – Analyse d'un en-tête TCP (ACK)

— Le numéro de séquence est ici `B9 6B 24 ABh`, soit un de plus par rapport au numéro de séquence du premier segment.

— Le numéro d'accusé de réception est `D5 64 9E 8Ch`, soit le numéro de séquence du segment précédent plus un. Le client a donc bien reçu tous les octets du segment envoyé par le serveur.

— La valeur de la suite de drapeaux est `10h`, soit `010000b` : les drapeaux `Urg`, `Psh`, `Rst`, `SYN` et `Fin` ne sont pas positionnés alors que le drapeau `Ack` l'est. Il indique que le numéro d'accusé de réception doit être pris en compte.

La connexion a donc bien été établie.

2.9 Un exemple d'application : HTTP

2.9.1 Le Web

L'application actuellement la plus utilisée sur Internet est le **Web**, plus exactement **WWW** (pour *World Wide Web*).

Notion d'hypertexte.- Lorsqu'on consulte une information, par exemple dans un dictionnaire, on a souvent besoin, de se référer à une autre information : la définition d'un mot inconnu renvoie à la définition d'autres mots inconnus. Qui n'a jamais rêvé, en consultant un dictionnaire papier, poser le doigt sur le mot inconnu de façon à ce que le dictionnaire s'ouvre automatiquement à la page sur laquelle se trouve sa définition, plutôt que d'avoir à trouver soi-même celle-ci par tâtonnements ?

S'il est difficile de la faire avec un dictionnaire papier, il est plus facile à mettre en œuvre ce mécanisme pour un texte numérique, en cliquant sur le mot avec la souris. C'est le principe de l'**hypertexte**.

Les protocoles du Web.- La mise en place de loin la plus utilisée de l'hypertexte (et même de l'**hypermédia**, puisqu'on peut faire apparaître ainsi, non seulement un texte, mais aussi une photo, un enregistrement sonore ou une vidéo) est le Web.

Le client bien connu pour se connecter à une page Web, puis aller d'une page à une autre, est le **navigateur** (Web), que nous n'allons pas décrire puisqu'il est devenu d'un usage courant.

Le Web repose sur deux protocoles :

— Le premier, **HTTP** (pour *HyperText Transfer Protocol*, défini en 1990 dans la RFC 1945), sert à demander une page Web sur un serveur et la recevoir.

— Une telle page Web est un simple fichier texte. La plupart du temps, il contient des données structurées suivant un second protocole : **HTML** (pour *HyperText Markup Language*, soit « langage de balisage d'hypertexte », défini en 1991 puis en 1995 dans la RFC 1866) qui permet une mise en page du texte et surtout contient des « ancres » pour aller à d'autres pages.

2.9.2 Le protocole HTTP

La communication entre le navigateur et le serveur Web se fait classiquement en deux temps : le navigateur effectue une *requête* HTTP ; le serveur traite la requête puis envoie une *réponse* HTTP.

2.9.2.1 Requête HTTP

Structure.- Une *requête HTTP* est un fichier texte, c'est-à-dire un ensemble de lignes. Elle comprend :

- Une *ligne de requête* : c'est une ligne précisant le document demandé, la méthode qui doit être appliquée, et la version du protocole utilisée. Cette ligne comprend trois champs devant être séparés par un espace :
  - la méthode ;
  - l'URL ;
  - la version du protocole utilisée par le client (généralement HTTP/1.0).
- Les *champs d'en-tête de la requête* est un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la requête et/ou le client (navigateur utilisé, système d'exploitation...). Chacune de ces lignes se compose d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête.
- Le *corps de la requête* est un ensemble de lignes optionnelles devant être séparées des lignes précédentes par une ligne vide.

Une requête HTTP a donc la syntaxe suivante (<crlf> signifiant retour chariot ou saut de ligne) :

```
METHODE URL VERSION<crlf>
EN-TETE : Valeur<crlf>
.
.
.
EN-TETE : Valeur<crlf>
Ligne vide<crlf>
CORPS DE LA REQUETE
```

Voici un exemple de requête HTTP :

```
GET http://www.commentcamarche.net HTTP/1.0
Accept : text/html
If-Modified-Since : Saturday, 15-January-2000 14:37:11 GMT
User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
```

Commandes.- La commande la plus importante est GET mais il y en a d'autres :

Commande	Description
GET	Requête de la ressource située à l'URL spécifiée
HEAD	Requête du seul en-tête de la ressource située à l'URL spécifiée
POST	Envoi de données au programme situé à l'URL spécifiée
PUT	Envoi de données à l'URL spécifiée
DELETE	Suppression de la ressource située à l'URL spécifiée

En-têtes.- Sans entrer dans les détails, voici quelques-uns des types d'en-têtes :

Nom de l'en-tête	Description
Accept	Type de contenu accepté par le navigateur (par exemple `text/html`)
Accept-Charset	Jeu de caractères attendu par le navigateur
Accept-Encoding	Codage de données accepté par le navigateur
Accept-Language	Langage attendu par le navigateur (anglais par défaut)
Authorization	Identification du navigateur auprès du serveur
Connection	Définit le type de connexion : `Close` : la connexion est fermée après la réponse `Keep-Alive` : crée une connexion persistante
Content-Encoding	Type de codage du corps de la requête
Content-Language	Type de langage du corps de la requête
Content-Length	Longueur du corps de la requête
Content-Type	Type de contenu du corps de la requête (par exemple `text/html`)
Cookie :	Permet d'envoyer au serveur les cookies qui ont été enregistrés On les écrit à la suite, séparés par un point-virgule nom-cookie=valeur cookie
Date	Date de début de transfert des données
Forwarded	Utilisé par les machines intermédiaires entre le navigateur et le serveur
From	Permet de spécifier l'adresse e-mail du client
From	Permet de spécifier que le document doit être envoyé s'il a été modifié depuis une certaine date
Host	Spécifie le nom de domaine du site, par exemple s'il y a plusieurs sites à la même adresse IP
Link	Relation entre deux URL
Orig-URL	URL d'origine de la requête
Referer	URL du lien à partir duquel la requête a été effectuée
User-Agent	Chaîne donnant des informations sur le client, comme le nom et la version du navigateur, du système d'exploitation

2.9.2.2 Réponse HTTP

Structure.- Une réponse HTTP est un fichier texte envoyé au navigateur par le serveur, constitué par :

- Une *ligne de statut*, qui est une ligne précisant la version du protocole utilisé et l'état du traitement de la requête à l'aide d'un code et d'un texte explicatif. Cette ligne comprend trois éléments devant être séparés par un espace :
  - la version du protocole utilisé ;
  - le code de statut ;
  - la signification du code.
- Les champs d'en-tête de la réponse : il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur. Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête.
- Le corps de la réponse, qui contient le document demandé.

Une réponse HTTP a donc la syntaxe suivante (<crlf> signifiant retour chariot ou saut de ligne) :

```
VERSION-HTTP CODE EXPLICATION<crlf>
EN-TETE : Valeur<crlf>
.
.
.
EN-TETE : Valeur<crlf>
Ligne vide<crlf>
CORPS DE LA REPONSE
```

Voici un exemple de réponse HTTP :

```
HTTP/1.0 200 OK
Date : Sat, 15 Jan 2000 14:37:12 GMT Server : Microsoft-IIS/2.0
Content-Type : text/HTML
Content-Length : 1245
Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT
```

En-têtes de réponse.- Sans entrer dans les détails, voici quelques-uns des types d'en-têtes de réponse :

Nom de l'en-tête	Description
Content-Encoding	Type de codage du corps de la réponse
Content-Language	Langue du corps de la réponse
Content-Length	Longueur du corps de la réponse
Content-Type	Type de contenu du corps de la réponse (par exemple `text/html`)
Date	Date de début de transfert des données
Expires	Date limite de consommation des données
Forwarded	Utilisé par les machines intermédiaires entre le navigateur et le serveur
Location	Redirection vers une nouvelle URL associée au document
Server	Caractéristiques du serveur ayant envoyé la réponse

Les codes de réponse.- Ce sont les codes affichés lorsque le navigateur n'arrive pas à fournir la page demandée. Le code de réponse est constitué de trois chiffres : le premier indique la classe de statut et les suivants la nature exacte de l'erreur.

Code	Message	Description
10x	Message d'information	Ces codes ne sont pas utilisés dans la version 1.0 du protocole
20x	Réussite	Ces codes indiquent le bon déroulement de la transaction
200	OK	La requête a été accomplie correctement
201	CREATED	Elle suit une commande POST, elle indique la réussite, le corps du reste du document est sensé indiquer l'URL à laquelle le document nouvellement créé devrait se trouver
202	ACCEPTED	La requête a été acceptée, mais la procédure qui suit n'a pas été accomplie
203	PARTIAL INFORMATION	Lorsque ce code est reçu en réponse à une commande GET, cela indique que la réponse n'est pas complète
204	NO RESPONSE	Le serveur a reçu la requête mais il n'y a pas d'information à renvoyer
205	RESET CONTENT	Le serveur indique au navigateur de supprimer le contenu des champs d'un formulaire
206	PARTIAL CONTENT	Il s'agit d'une réponse à une requête comportant l'en-tête `range`. Le serveur doit indiquer l'en-tête `content-Range`
30x	Redirection	Ces codes indiquent que la ressource n'est plus à l'emplacement indiqué
301	MOVED	Les données demandées ont été transférées à une nouvelle adresse
302	FOUND	Les données demandées sont à une nouvelle URL, mais ont cependant peut-être été délacées depuis
303	METHOD	Cela implique que le client doit essayer une nouvelle adresse, en essayant de préférence une autre méthode que GET
304	NOT MODIFIED	Si le client a effectué une commande GET conditionnelle (en demandant si le document a été modifié depuis la dernire fois) et que le document n'a pas été modifi, le serveur renvoie ce code
40x	Erreur due au client	Ces codes indiquent que la requête est incorrecte
400	BAD REQUEST	La syntaxe de la requête est mal formulée ou est impossible à satisfaire
401	UNAUTHORIZED	Le paramètre du message donne les spécifications des formes d'autorisation acceptables. Le client doit reformuler sa requête avec les bonnes données d'autorisation
402	PAYMENT REQUIRED	Le client doit reformuler sa demande avec les bonnes données de paiement
403	FORBIDDEN	L'accès à la ressource est tout simplement interdit
404	NOT FOUND	Le serveur n'a rien trouvé à l'adresse spécifiée
50x	Erreur due au serveur	Ces codes indiquent qu'il y a eu une erreur interne du serveur
500	INTERNAL ERROR	Le serveur a rencontré une condition inattendue qui l'a empêché de donner suite à la demande
501	NOT IMPLEMENTED	Le serveur ne supporte pas le service demandé
502	BAD GATEWAY	Le serveur a reçu une réponse non valide de la part du serveur auquel il essayait d'accéder en agissant comme une passerelle ou un proxy
503	SERVICE UNAVAILABLE	Le serveur ne peut pas vous répondre à l'instant présent, car le trafic est trop dense
504	GATEWAY TIMEOUT	La réponse du serveur n'est pas parvenue dans les temps impartis

2.9.3 Analyse d'une requête HTTP

Récupérons avec *Wireshark* une trame HTTP (figure 2.19) de taille relativement courte (avec comme paramtres HTTP pour le protocole et GET comme info). Analysons cette trame complètement, en se souvenant que seul le message nous importe vraiment :

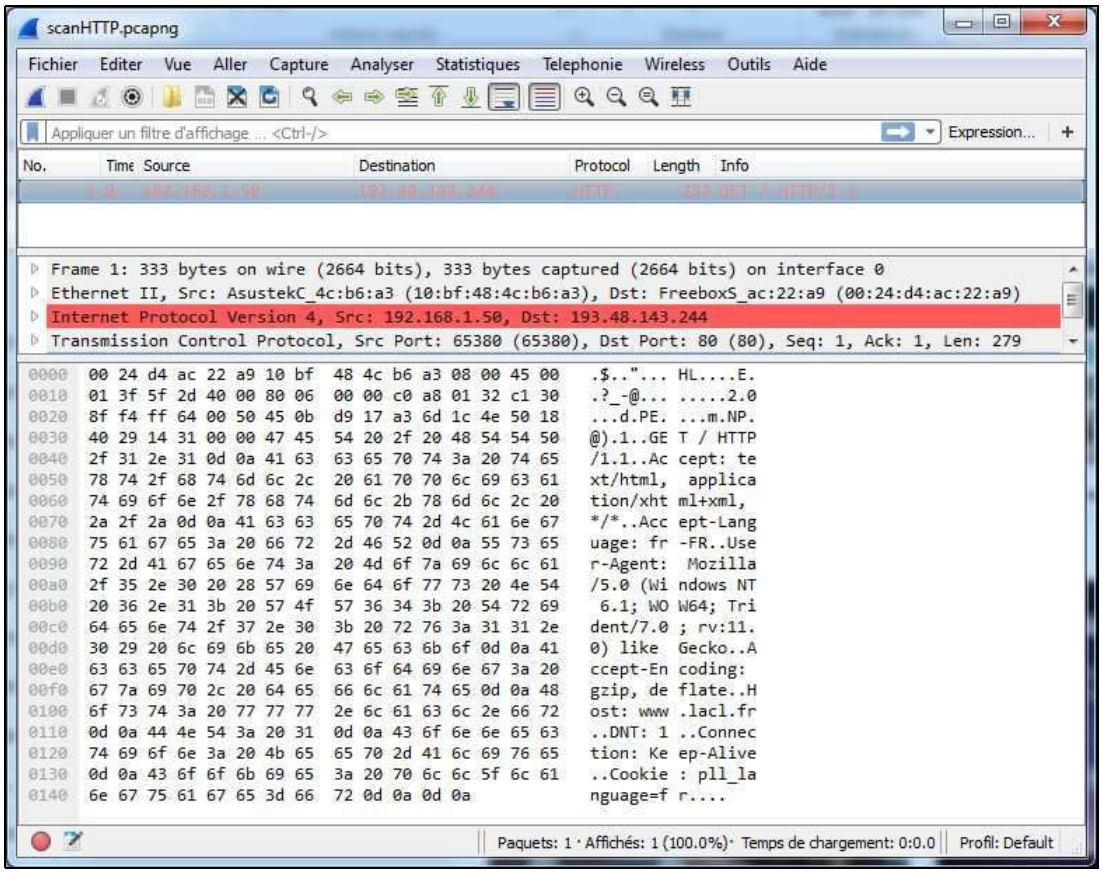

FIGURE 2.19 – Analyse d'une requête HTTP

Analyse de l'en-tête Ethernet.- Cette trame de 333 octets provient d'une interface Ethernet donc commence par un en-tête Ethernet (sans préambule) :

- Les six premiers octets donnent l'adresse MAC du destinataire : `00:24:d4:ac:22:a9`.
- Les six octets suivants donnent l'adresse MAC de l'expéditeur : `10:bf:48:4c:b6:a3`.
- les deux octets suivants, 08 00h, soit 2048 donne le type, donc IP.

Analyse de l'en-tête IPv4.- Puisque le type Ethernet est IP, cet en-tête est suivi d'un en-tête IP, certainement IPv4 :

- Le premier demi-octet, 4, donne la version d'IP. Il s'agit bien d'IPv4.
- Le second demi-octet, 5, donne la longueur de l'en-tête IP en mots de 4 octets. On a donc 20 octets, soit le minimum pour un en-tête IPv4. Il n'y a pas d'options.
- L'octet suivant donne le type de service. Il est nul, ce qui est habituel pour une extrémité.
- Les deux octets suivants, 0x01 3f, donnent la longueur du paquet, soit 319 octets. Avec

les 14 octets de l'ent-tête Ethernet, on retrouve bien les 333 octets de la trame ; il n'y a donc rien d'autre que le paquet IPv4 comme données de la trame.
- Les deux octets suivants, 0x5f 2d, donnent l'identificateur du paquet, sur lequel il n'y a rien de plus à dire puisqu'il est choisi presqu'au hasard.
- Les deux octets suivants, 0x40 00, représentent deux champs :
  - Les premiers bits 010b sont des drapeaux : le premier est « réservé » donc sa valeur nulle n'est pas étonnante ; le second DF, qui est levé, indique que le paquet originel ne doit pas être fragmenté ; le dernier MF, qui n'est pas levé, indique qu'il n'y a pas de fragment sui suit, ce qui est normal puisque le paquet originel ne devait pas être fragmenté.
  - Les treize derniers bits donnent le décalage par rapport au paquet originel. Il est nul, ce qui est normal puisque le paquet originel ne devait pas être fragmenté.
- L'octet suivant 80h, donne la durée de vie, c'est-à dire le nombre de sauts que peut traverser le paquet avant d'être écart : la valeur 128 est une valeur courante pour un paquet en partance.
- L'octet suivant, 06h, donne le « protocole » : il s'agit ici de TCP.
- Les deux octets suivants donnent la somme de contrôle. Elle est nulle car elle n'a pas encore été calculée par le module IP.
- Les quatre octets suivants, c0 a8 01 32h, donnent l'adresse IP de l'expditeur : 192.168.1.50. il s'agit d'une adresse privée donnée dynamiquement par le routeur du réseau local.
- Les quatre octets suivants, c1 30 8f f4h, donnent l'adresse IP du destinataire : 193.48.143.244.

Analyse de l'en-tête TCP.- Puisque le protocole IP est TCP, l'en-tête IPv4 est suivi d'un en-tête TCP :

- Les deux premiers octets, ff 64h, donnent le port de l'expéditeur. Il est supérieur à 2 048, il s'agit donc certainement d'un port client donné de façon presqu'aléatoire par le module TCP.
- Les deux octets suivants, 00 50h, donnent le port du destinataire. Il s'agit d'un port bien connu : 80 pour un service HTTP.
- Les 4 octets suivants, 45 0b d9 17h, donnent le numéro de séquence.
- Les 4 octets suivants, a3 6d 1c 4eh, donnent le numéro d'accusé de réception.
- Les deux octets suivants, 50 18h, représentent deux champs :
  - Le demi-octet suivant, 5h, donne la longeur de l'en-tête TCP en mots de 4 octets. On a donc 20 octets, ce qui est la taille minimum. Il n'y a donc pas d'options.
  - Les 12 bits suivants , 0000 0001 1000b, sont des drapeaux : les six premiers sont à l'origine réservés, donc nuls ; le drapeau urgent URG n'est pas levé ; le drapeau d'accusé de réception ACK est levé, ce qui indique qu'il faut prendre en compte le numéro d'accusé de réception ; le drapeau PuSH est levé, c'est donc la fin du message qu'il faudra envoyer à l'application ; les drapeaux ReSeT, SYn et FIN ne sont pas levés.
- Les deux octets suivants, 40 29h, donnent la taille de la fenêtre : 16 425 donc le débit peut être élevé.
- Les deux octets suivants, 14 31h, donnent la somme de contrôle.
- Les deux octets suivants, 00 00h, donnent la valeur du pointeur urgent, qui est nul, ce qui est normal puisque le drapeau urgent n'est pas levé.

Analyse du message HTTP.- Nous sommes arrivés à la fin des 20 octets de l'en-tête TCP, la suite est donc constituée du message HTTP.

Puisqu'un message HTTP est un fichier texte, nous avons intérêt à utiliser la fenêtre ASCII de l'éditeur hexadécimal plutôt que la fenêtre hexadécimale, sauf lorsque c'est nécessaire. Cette

fenêtre ASCII donne, pour chaque octet, la représentation du caractère ASCII dont le code est cet octet, tout au moins pour les caractères « affichables », les autres caractères étant affichés comme des points « . ».

Rappelons, pour les caractères non affichables, qui nous intéressent que : 0dh = 13 est le retour chariot noté « \r », c'est-à-dire le retour en début de ligne ; 0ah = 10 est le passage à la ligne suivante noté « \n ».

Pour le système d'exploitation *Windows*, un passage à la ligne demande deux caractères ASCII : le retour chariot suivi du passage à la ligne suivante.

Le message HTTP est donc :

```
GET / HTTP/1.1\r\n
Accept: text/html, application/xhtml+xml, */*\r\n
Accept-Language: fr-FR\r\n
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko\r\n
Accept-Encoding: gzip, deflate\r\n
Host: www.lacl.fr\r\n
DNT: 1\r\n
Connection: Keep-Alive\r\n
Cookie: pll_language=fr\r\n
\r\n
```

Il s'agit d'une requête, demandant le fichier se trouvant à la racine du site.

Commentaires.- 1^o) Le type de contenu accepté par le navigateur (en-tête `Accept`) est un type **MIME** (*Multipurpose Internet Mail Extensions*), standard proposé par les laboratoires *Bell Communications* en 1991 afin d'étendre les possibilités limitées du courrier électronique, constitué alors uniquement par du texte ASCII, et notamment de permettre d'insérer des documents (images, sons...) dans un courrier. Il est défini à l'origine par les RFC 1341 et 1342 datant de juin 1992. Une liste officielle des types MIME est maintenue par IANA :

`http://www.iana.org/assignments/media-types/media-types.xhtml`

2^o) Le navigateur utilisé pour formuler la requête est *Internet Explorer*. Remarquez sa description dans l'en-tête `User-Agent`.

3^o) On peut compresser la réponse (en-tête `Accept-Encoding`) grâce au logiciel libre de compression `gzip` (acronyme de *GNU zip*), créé en 1991 pour remplacer le programme `compress` d'Unix, en utilisant la méthode de compression `deflate`.

4^o) En plus d'être donné par son adresse IP dans l'en-tête IPv4, le site recherché est donné par son nom de domaine dans l'en-tête `Host` : `www.lacl.fr`.

5^o) L'en-tête `DNT` (*Do Not Track*) est un projet, datant de 2009, qui permet de signaler aux applications web que l'utilisateur ne veut pas être « suivi ». Ce n'est qu'un signalement fait par l'utilisateur.

6^o) *Polylang* utilise un cookie (`pll-cookie`) pour rappeler la langue sélectionnée par l'utilisateur lorsqu'il revient visiter un site web.

2.9.4 Analyse d'une réponse HTTP

Le problème avec les réponses HTTP est qu'elles sont devenues fort volumineuses. La meilleure solution est certainement de créer notre propre serveur HTTP (en Java, par exemple, avec un port « expérimental » différent de 80 si celui-ci est déjà utilisé). Ici nous allons récupérer une trame avec *Wireshark* en faisant appel à :

`http://www.lacl.fr/cegielski/`

ce qui donne une réponse correspondant à la figure 2.20).

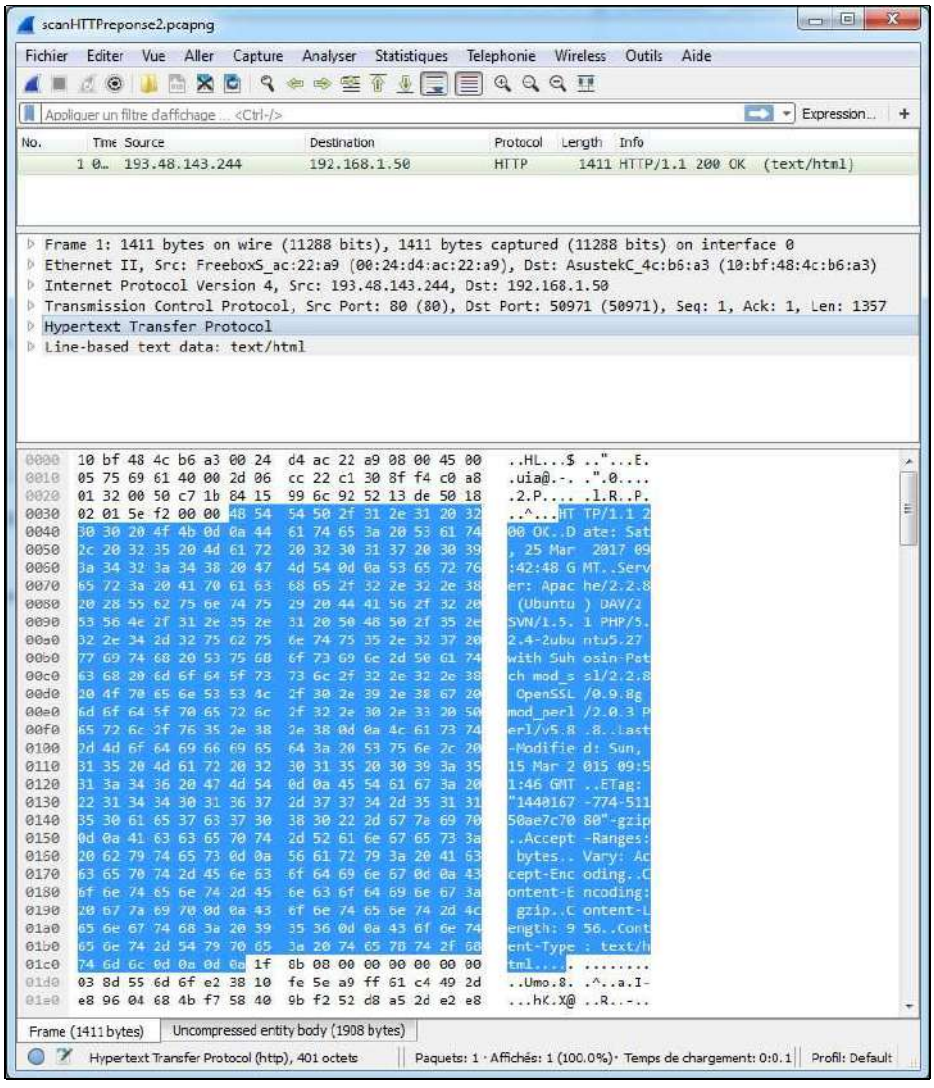

FIGURE 2.20 – Analyse d'une réponse HTTP

Analyse de l'en-tête Ethernet.- Cette trame de 1 411 octets provient d'une interface Ethernet donc commence par un en-tête Ethernet (sans préambule) :

— Les six premiers octets donnent l'adresse MAC du destinataire : `10:bf:48:4c:b6:a3`.

— Les six octets suivants donnent l'adresse MAC de l'expéditeur : `00:24:d4:ac:22:a9`. Les adresses MAC sont donc échangées par rapport à la requête, ce qui est normal.
— les deux octets suivants, 08 00h, soit 2048 donne le type, donc IP.

Analyse de l'en-tête IPv4.- Puisque le type Ethernet est IP, cet en-tête est suivi d'un en-tête IP, certainement IPv4 :

— Le premier demi-octet, 4, donne la version d'IP. Il s'agit bien d'IPv4.
— Le second demi-octet, 5, donne la longueur de l'en-tête IP en mots de 4 octets. On a donc 20 octets, soit le minimum pour un en-tête IPv4. Il n'y a pas d'options.
— L'octet suivant donne le type de service. Il est nul, ce qui est habituel pour une extrémité.
— Les deux octets suivants, 0x05 75, donnent la longueur du paquet, soit 1 397 octets. Avec les 14 octets de l'ent-tête Ethernet, on retrouve bien les 1 411 octets de la trame ; il n'y a donc rien d'autre que le paquet IPv4 comme données de la trame.
— Les deux octets suivants, 0x06 61, donnent l'identificateur du paquet, sur lequel il n'y a rien de plus à dire puisqu'il est choisi presqu'au hasard.
— Les deux octets suivants, 0x40 00, représentent deux champs :
  — Les premiers bits 010b sont des drapeaux : le premier est « réservé » donc sa valeur nulle n'est pas étonnante ; le second DF, qui est levé, indique que le paquet originel ne doit pas être fragmenté ; le dernier MF, qui n'est pas levé, indique qu'il n'y a pas de fragment sui suit, ce qui est normal puisque le paquet originel ne devait pas être fragmenté.
  — Les treize derniers bits donnent le décalage par rapport au paquet originel. Il est nul, ce qui est normal puisque le paquet originel ne devait pas être fragmenté.
— L'octet suivant 0x2d, donne la durée de vie, c'est-à dire le nombre de sauts que peut traverser le paquet avant d'être écart : la valeur 45 montre qu'il a traversé un certain nombre de sauts avant de nous parvenir.
— L'octet suivant, 06h, donne le « protocole » : il s'agit ici de TCP.
— Les deux octets suivants donnent la somme de contrôle, cc 22h. Nous vous laissons vérifier qu'elle est correcte.
— Les quatre octets suivants, c1 30 8f f4h, donnent l'adresse IP de l'expditeur : 193.48.143.244. il s'agit d'une adresse privée donnée dynamiquement par le routeur du réseau local.
— Les quatre octets suivants, c0 a8 01 32h, donnent l'adresse IP du destinataire : 192.168.1.50. Comme pour les adresses MAC, elles sont échangées par rapport à la requête, ce qui est normal.

Analyse de l'en-tête TCP.- Puisque le protocole IP est TCP, l'en-tête IPv4 est suivi d'un en-tête TCP :

— Les deux premiers octets, 00 50h, donnent le port de l'expéditeur. Il s'agit d'un port bien connu : 80 pour un service HTTP.
— Les deux octets suivants, 0xc7 1b, donnent le port du destinataire. Il est supérieur à 2 048, il s'agit donc certainement d'un port client donné de façon presqu'aléatoire par le module TCP.

  Il ne correspond pas au port de la requête étudiée ci-dessus, car il ne s'agit pas de la réponse à cette requête.
— Les 4 octets suivants, 0x84 15 99 6c, donnent le numéro de séquence.
— Les 4 octets suivants, 0x92 52 13 de, donnent le numéro d'accusé de réception.
— Les deux octets suivants, 50 18h, représentent deux champs :
  — Le demi-octet suivant, 5h, donne la longeur de l'en-tête TCP en mots de 4 octets. On a donc 20 octets, ce qui est la taille minimum. Il n'y a donc pas d'options.

— Les 12 bits suivants , 0000 0001 1000b, sont des drapeaux : les six premiers sont à l'origine réservés, donc nuls ; le drapeau urgent URG n'est pas levé ; le drapeau d'accusé de réception ACK est levé, ce qui indique qu'il faut prendre en compte le numéro d'accusé de réception ; le drapeau PuSH est levé, c'est donc la fin du message qu'il faudra envoyer à l'application ; les drapeaux ReSeT, SYn et FIN ne sont pas levés.
— Les deux octets suivants, 02 01h, donnent la taille de la fenêtre : 513 donc le débit est moins élevé que pour la requête.
— Les deux octets suivants, 5e f2h, donnent la somme de contrôle.
— Les deux octets suivants, 00 00h, donnent la valeur du pointeur urgent, qui est nul, ce qui est normal puisque le drapeau urgent n'est pas levé.

Analyse du message HTTP.- Nous sommes arrivés à la fin des 20 octets de l'en-tête TCP, la suite est donc constituée du message HTTP.

Comme pour la requête HTTP, nous avons intérêt à utiliser la fenêtre ASCII de l'éditeur hexadécimal.

Le message HTTP est donc :

```
HTTP/1.1 200 OK\r\n
Date: Sat,25 Mar 2017 09:42:48 GMT\r\n
Server: Apache/2.2.8 (Ubuntu) DAV/2 SVN/1.5.1 PHP/5.2.4-2ubuntu5.27 with
   Suhsoin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g mod_perl/2.0.3 Perl/v5.8.8\r\n
Last-Modified: Sun, 15 Mar 2015 09:51:46 GMT\r\n
ETag: "1440167-774-51150ae7c7080"-gzip\r\n
Accept-Ranges: bytes\r\n
Vary: Accept-Encoding\r\n
Content-Encoding: gzip\r\n
Content-Length: 956\r\n
Content-Type: text/html\r\n
\r\n
...
```

Il s'agit d'une réponse.

Commentaires.- 1^o) Nous sommes dans le cas de réponse le plus favorable : 200 OK.

2^o) La date à laquelle la réponse est envoyée est donnée à la seconde près. Il ne s'agit pas de la date locale du destinataire, ou de l'expéditeur, mais du temps GMT.

3^o) Les caractéristiques du serveur sont décrites avec beaucoup de détails.

4^o) La date de dernière modification de la page permet au navigateur de ne pas tout récupérer à nouveau s'il le désire.

5^o) La balise-entité `ETag`, ou étiquette d'entité, est utilisée pour la validation du cache. Un ETag est un identifiant unique opaque assigné par le serveur web à chaque version d'une ressource accessible via une URL.

6^o) L'en-tête `Accept-Ranges` est utilisé par le serveur pour dire qu'il accepte des requêtes partielles (pour reprendre une téléchargement interrompu). Les deux valeurs possibles sont `none` et `bytes`.

7^o) Il y a eu un choix par rapport aux méthodes de compression proposées. La réponse compresse les données du message en utilisant `gzip`.

8^o) La longueur des données est de 956 octets, ce qui correspond au message moins les en-têtes.

9^o) Le type de contenu est les plus simple, à savoir `text/html`.

10^o) Il est difficile de visualiser les données puisqu'elles sont compressées par `gzip`. On peut les copier et utiliser un logiciel pour les voir.

Wireshark nous aide en le faisant pour nous. Donnons-en le tout début :

```
<html>\r\n
\r\n
<head>\r\n
\r\n
<body BGCOLOR="#FFFFFF" LINK=blue ALINK=red VLINK=green>\r\n
\r\n
<img align=right src="PCegielski.jpg" WIDTH=300 HEIGHT=466>\r\n
...
```

2.10 Le protocole ICMP

Les messages ICMP sont utilisés pour indiquer des erreurs d'un hôte à l'autre ou d'un hôte à une passerelle (*gateway* en anglais). De passerelle à passerelle, on utilise un autre protocole : **GGP** pour *Gateway to Gateway Protocol*.

Les messages ICMP sont envoyés au-dessus de IP, avec un en-tête IP dont le type de protocole est 1, comme nous l'avons déjà vu.

2.10.1 En-tête des unités d'information ICMP

L'en-tête des unités d'information ICMP comprend 32 bits :

Type (8 bits)	Code (8 bits)	Somme de contrôle (16 bits)

— Le champ `type` d'un octet spécifie le type de message d'erreur ou de réponse tel que *destination unreachable*, *echo*, *echo reply* ou *redirect message*.

— Le champ `code` d'un octet spécifie des informations supplémentaires si nécessaire.

Par exemple si le message est du type *destination unreachable*, il existe plusieurs valeurs possibles pour ce code telles que *network unreachable*, *host unreachable* ou *port unreachable*.

La liste complète des types et des codes des messages ICMP est tenue à jour sur le site Web de l'IANA :

`http://www.iana.com`

— Le champ `somme de contrôle` (*checksum* en anglais) porte sur le message en entier, y compris l'en-tête IP.

2.10.2 Quelques messages ICMP

Les messages ICMP se répartissent en trois catégories : les messages d'erreur (destinataire inaccessible, temps écoulé qui informent les hôtes des problèmes se révélant au cours de leurs envois ; il ne s'agit pas d'une correction d'erreur, puisque les messages ne sont pas retransmis mais d'un système d'avertissement d'erreur), de contrôle (ralentissement de la source) et d'analyse de réseau (l'écho).

2.10.2.1 Requête et réponse d'écho (types 8 et 0)

Un administrateur réseau souhaitant vérifier si un hôte donné est toujours opérationnel peut lui envoyer un message d'écho (type 8, sans code). Si l'hôte est actif, il lui renvoie une réponse d'écho (type 0, sans code).

L'utilitaire le plus courant pour l'envoi d'un message d'écho et l'analyse des réponses s'appelle **ping** (*Packet INternet Groper*).

Type (8 bits)	0	Somme de contrôle (16 bits)
Identificateur		Numéro de séquence
Données		

— Comme nous l'avons vu, le type est 8 pour une demande d'écho et 0 pour une réponse.

— L'identificateur (*identifier* en anglais) est décidé lors de la requête. Il est renvoyé lors de la réponse.

Type	Code	Description	Requête	Erreur	Référence
0	0	Echo Reply	x		RFC 792
3	0	Network unreachable		x	RFC 792
3	1	Host unreachable		x	RFC 792
3	2	Protocol unreachable		x	RFC 792
3	3	Port unreachable		x	RFC 792
3	4	Fragmentation nécessaire mais drapeau Don't F		x	RFC 792
3	5	Source routing failed		x	RFC 792
3	6	Destination network unknown		x	RFC 792
3	7	Destination host unknown		x	RFC 792
3	8	Source host isolateted (obsolete)		x	RFC 792
3	9	Destination network adminsitratively prohibited		x	RFC 792
3	10	Destination host adminsitratively prohibited		x	RFC 792
3	11	Destination unreachable for TOS		x	RFC 792
3	12	Host unreachable for TOS		x	RFC 792
3	13	Communication adminsitratively prohibited by filtering		x	RFC 1812
3	14	Host precedence violation		x	RFC 1812
3	15	Precedence cutoff in effect		x	RFC 1812
4	0	Source quench			RFC 792
5	0	Redirect for network			RFC 792
5	1	Redirect for host			RFC 792
5	2	Redirect for TOS and network			RFC 792
5	3	Redirect for TOS and host			RFC 792
8	0	Echo request	x		RFC 792
9	0	Router advertisement - Normal			RFC 1256
9	16	Router advertisement - Does not route common traffic			RFC 2002
10	0	Route selection			RFC 1256
11	0	TTL equals 0 during transit		x	RFC 792
11	1	TTL equals 0 during reassembly		x	RFC 792
12	0	IP header bad		x	RFC 792
12	1	Required options missing		x	RFC 792
12	2	IP header bad length		x	RFC 792
13	0	Timestamp request (obsolete)	x		RFC 792
14		Timestamp reply (obsolete)	x		RFC 792
15	0	Information request (obsolete)	x		RFC 792
16	0	Information reply (obsolete)	x		RFC 792
18	0	Address mask reply	x		RFC 950
30	0	Traceroute	x		RFC 1393
31	0	Datagram conversion Error		x	RFC 1475

FIGURE 2.21 – Types et codes ICMP

— Le numéro de séquence (*Sequence Number* en anglais) est décidé par chaque hôte.
— Par défaut le champ des données (*Data* en anglais) est vide mais il peut contenir une quantité spécifiée par l'utilisateur de données aléatoires.

2.10.2.2 Destination inaccessible (type 3)

La principale famille de messages d'erreur ICMP regroupe les messages de destination inaccessible (type 3), émis par les routeurs ne parvenant pas à transmettre les paquets qu'ils traitenet à leurs destinataires.

3	Code (8 bits)	Somme de contrôle (16 bits)
Non utilisé		
En-tête Internet + 64 bits du datagramme d'origine		

— Le type *Destination Unreachable* a 16 codes de base :
  — Code 0 (*Network unreachable*) dit que le réseau (local du destinataire) n'est pas accessible actuellement.
  — Code 1 (*Host unreachable*) dit que (le réseau est accessible mais que) l'hôte n'est pas accessible actuellement.
  — Code 2 (*Protocol unreachable*) dit que le protocole (TCP, UDP, etc) n'est pas utilisable actuellement.
  — Code 3 (*Port unreachable*) dit que le port n'est pas accessible actuellement.
  — Code 4 (*Fragmentation needed and DF set*) dit qu'un paquet doit être fragmenté mais que le drapeau *Do not fragment* est présent, d'où ce retour de la part de la passerelle.
  — Code 5 (*Source route failed*) dit que le routage a échoué.
  — Code 6 (*Destination network unknown*) dit qu'il n'existe pas de route vers le réseau indiqué.
  — Code 7 (*Destination host unknown*) dit qu'il n'existe pas de route vers l'hôte indiqué.
  — Code 8 (*Source host isolated*) doit être renvoyé si l'hôte est isolé. Ce code est maintenant obsolète.
  — Code 9 (*Destination network administratively prohibited*) dit que le réseau est bloqué à une passerelle.
  — Code 10 (*Destination host administatively prohibited*) dit qu'on ne peut pas accéder à l'hôte à cause de l'administration du routage.
  — Code 11 (*Network unreachable for TOS*) dit que le réseau n'est pas accessible à cause d'un mauvais positionnement du TOS du paquet.
  — Code 12 (*Host unreachable for TOS*) dit que l'hôte n'est pas accessible à cause du TOS du paquet.
  — Code 13 (*Communication administratively prohibited by filtering*) dit que le paquet a été interdit à cause du filtrage, par exemple à un pare-feu.
  — Code 14 (*Host precedence violation*) est envoyé par le premier routeur pour notifier à un hôte connecté que la priorité utilisée n'est pas permise pour cette combinaison destination/source.
  — Code 15 (*Precedence cutoff in effect*) est envoyé par le premier routeur si le paquet a une priorité trop basse.
— Le message contient une petite partie des données, contenant l'entier IP complet et 64 bits du paquet IP originel, qui devrait comporter les ports.

2.10.2.3 Ralentissement de la source (type 4)

Certains types de messages ICMP servent à altérer le fonctionnement des hôtes ou routeurs du réseau. Un message de demande de ralentissement de la source (*Source Quench* en anglais ; type 4, sans code) sert au **controle de flux**, c'est-à-dire à gérer le débit auquel les hôtes envoient leurs paquets. Un hôte recvant ce type de message est invité à réduire son débit de transmission.

4	0	Somme de contrôle (16 bits)
Non utilisé		
En-tête Internet + 64 bits du datagramme d'origine		

2.10.2.4 Redirection (type 5)

Supposons que vous ayez un réseau (192.168.0.0/24) comportant plusieurs hôtes et deux passerelles : une passerelle vers le réseau 10.0.0.0/24 et une passerelle par défaut pour le reste. Considérons qu'un des hôtes du réseau 192.168.0.0/24 soit paramétré pour envoyer *via* la passerelle par défaut. S'il envoie un paquet à 10.0.0.26, celui-ci passera par la passerelle par défaut, qui connaît bien sûr le réseau 10.0.0.0/24. Mais la passerelle par défaut sait qu'il est plus rapide d'envoyer le paquet directement à la passerelle 10.0.0.0/24. Elle enverra donc un message ICMP *Redirect* à l'hôte, en lui disant la bonne passerelle à utiliser tout en envoyant son paquet vers la passerelle 10.0.0.0/24. L'hôte connaîtra maintenant l'existence de cette passerelle et espérons qu'il l'utilisera dans le futur.

5	Code (8 bits)	Somme de contrôle (16 bits)
Adresse Internet de la passerelle		
En-tête Internet + 64 bits du datagramme d'origine		

- Le champ principal est celui de l'adresse Internet de la passerelle à utiliser.
- Il y a 4 codes possibles :
  - Code 0 (*Redirect for network*) est seulement utilisé pour rediriger vers un réseau en entier (comme dans l'exemple ci-dessus).
  - Code 1 (*Redirect for host*) est seulement utilisé pour rediriger vers un hôte donné.
  - Code 2 (*Redirect for TOS and network*) est seulement utilisé pour rediriger dans le cas d'un type de service et d'un réseau complet donnés.
  - Code 3 (*Redirect for TOS and host*) est seulement utilisé pour rediriger dans le cas d'un type de service et d'un hôte donnés.

2.10.2.5 Temps écoulé (type 11)

Le message ICMP *temps écoulé* (*TTL equals 0* ou *Times Exceeded* en anglais) est envoyé à l'hôte si le champ `TTL` atteint 0 lors du transit dans une passerelle ou lors du réassemblage des fragments sur l'hôte de destination, tout en écartant le paquet.

11	Code (8 bits)	Somme de contrôle (16 bits)
non utilisé		
En-tête Internet + 64 bits du datagramme d'origine		

Deux codes sont utilisés :

— Code 0 (*TTL equals 0 during transit*) est envoyé si le TTL a atteint 0 lors du transit par une passerelle.

— Code 1 (*TTL equals 0 during reassembly*) est envoyé si le TTL a atteint 0 lors du réassemblage des fragments.

2.10.2.6 Problème de paramétrage

Le message ICMP *Parameter problem* est utilisé pour dire à l'expéditeur qu'une passerelle ou l'hôte de destination a des problèmes à comprendre certaines parties de l'en-tête IP.

Type (8 bits)	Code (8 bits)	Somme de contrôle (16 bits)
Pointeur	non utilisé	
En-tête Internet + 64 bits du datagramme d'origine		

— Deux codes sont utilisés :

  — Code 0 (*IP header bad (catchall error)*) est envoyé s'il y a une erreur dans l'en-tête IP. Le champ *pointeur* spécifie la partie de l'en-tête qui pose problème.

  — Code 1 (*Required options missing*) est envoyé si une option requise manque.

— Le champ pointeur (*pointer*) sert dans le cas du code 0.

2.10.3 Analyse de trames ICMP

Effectuons un ping (voir figure 2.20) tout en capturant les trames grâce à `Wireshark`. Nous obtenons un certain nombres de trames de reqêtes d'écho et de réponses d'écho.

FIGURE 2.22 – L'utilitaire ping

2.10.3.1 Requête d'écho

Étudions la première trame, qui est une requête d'écho (voir figure 2.21). Il n'y a pas grand chose à dire sur les en-têtes Ethernet et IP, sinon que le protocole de ce dernier en-tête est `01h`, soit 1, pour indiquer un message ICMP. Commentons plus en détail ce message ICMP.

— Le type est `08h`, ce qui indique une requête d'écho.

— Le code est `00h` ; il n'y a pas d'autre valeur possible pour une requête d'écho.

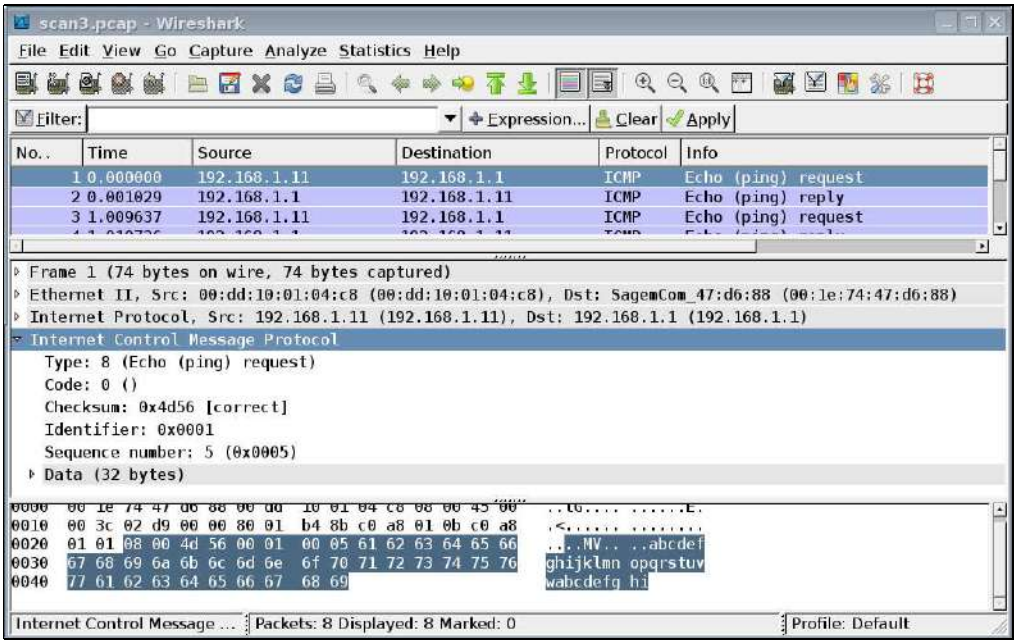

FIGURE 2.23 – Requête d'écho

— La somme de contrôle est `4D56h`. `Wireshark` dit que celle-ci est correcte ; nous laissons le soin au lecteur de vérifier.

— L'identificateur est `0001h`, soit 1.

— Le numéro de séquence est `0005h`, soit 5.

— Il y a 32 octets de données, constitués tout simplement du code ASCII des seize premières lettres minuscules de l'alphabet.

2.10.3.2 Reponse d'écho

Étudions également la seconde trame, qui est la réponse à notre requête d'écho (voir figure 2.22). Il n'y a toujours pas grand chose à dire sur les en-têtes Ethernet et IP, sinon que le protocole de ce dernier en-tête est également `01h`, soit 1, pour indiquer un message ICMP. Commentons plus en détail ce message ICMP.

— Le type est `00h`, ce qui indique une réponse d'écho.

— Le code est `00h` ; il n'y a pas d'autre valeur possible pour une requête d'écho.

— La somme de contrôle est `5556h`. `Wireshark` dit que celle-ci est correcte ; nous laissons le soin au lecteur de vérifier.

— L'identificateur est `0001h`, soit 1, ce qui est bien le même que celui de la requête.

— Le numéro de séquence est `0005h`, soit 5.

— Il y a 32 octets de données, constitués tout simplement du code ASCII des seize premières lettres minuscules de l'alphabet.

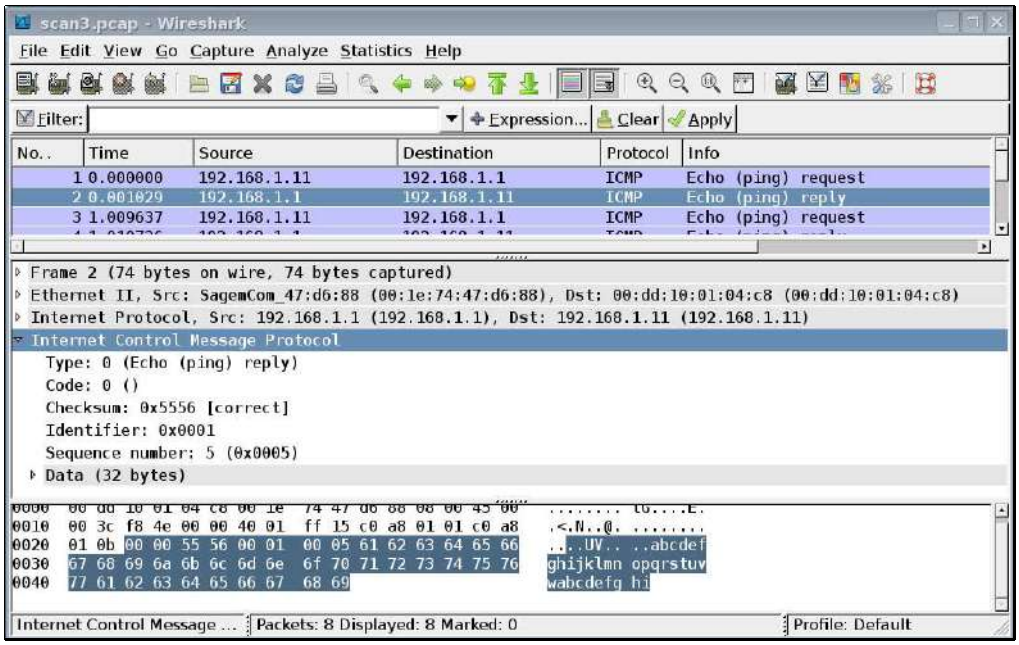

FIGURE 2.24 – Réponse d'écho